

Application Note

78K0/Kx1+

8-Bit Single-Chip Microcontrollers

Flash Memory Programming (Programmer)

μ PD78F0101H

μ PD78F0102H

μ PD78F0103H

μ PD78F0112H

μ PD78F0113H

μ PD78F0114H

μ PD78F0114HD

μ PD78F0122H

μ PD78F0123H

μ PD78F0124H

μ PD78F0124HD

μ PD78F0132H

μ PD78F0133H

μ PD78F0134H

μ PD78F0136H

μ PD78F0138H

μ PD78F0138HD

μ PD78F0148H

μ PD78F0148HD

[MEMO]

NOTES FOR CMOS DEVICES

① VOLTAGE APPLICATION WAVEFORM AT INPUT PIN

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (MAX) and V_{IH} (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (MAX) and V_{IH} (MIN).

② HANDLING OF UNUSED INPUT PINS

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

③ PRECAUTION AGAINST ESD

A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

④ STATUS BEFORE INITIALIZATION

Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

⑤ POWER ON/OFF SEQUENCE

In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current.

The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

⑥ INPUT OF SIGNAL DURING POWER OFF STATE

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

• **The information in this document is current as of September, 2005. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**

- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

"Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.

"Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).

"Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

- (1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
- (2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

INTRODUCTION

Target Readers	This application note is intended for users who understand the functions of the 78K0/Kx1+ and who will use this product to design application systems.																
Purpose	<p>The purpose of this application note is to help users understand how to develop dedicated flash memory programmers for rewriting the internal flash memory of the 78K0/Kx1+.</p> <p>The sample programs and circuit diagrams shown in this document are for reference only and are not intended for use in actual design-ins.</p> <p>Therefore, these sample programs must be used at the user's own risk. Correct operation is not guaranteed if these sample programs are used.</p>																
Organization	<p>This manual consists of the following main sections.</p> <ul style="list-style-type: none">• Flash memory programming• Programmer operating environment• Basic programmer operation• Command/data frame format• Description of command processing• UART communication mode• 3-wire serial I/O communication mode with handshake supported (CSI + HS)• 3-wire serial I/O communication mode (CSI)• Flash memory programming parameter characteristics• Electrical specifications (reference)																
How to Read This Manual	<p>It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.</p> <p><input type="checkbox"/> To learn more about the 78K0/Kx1+'s hardware functions: → See the user's manual of each 78K0/Kx1+ product.</p>																
Conventions	<table><tr><td>Data significance:</td><td>Higher digits on the left and lower digits on the right</td></tr><tr><td>Active low representation:</td><td>\overline{xxx} (overscore over pin or signal name)</td></tr><tr><td>Note:</td><td>Footnote for item marked with Note in the text</td></tr><tr><td>Caution:</td><td>Information requiring particular attention</td></tr><tr><td>Remark:</td><td>Supplementary information</td></tr><tr><td>Numeral representation:</td><td>Binaryxxxx or xxxxB</td></tr><tr><td></td><td>Decimalxxxx</td></tr><tr><td></td><td>HexadecimalxxxxH</td></tr></table>	Data significance:	Higher digits on the left and lower digits on the right	Active low representation:	\overline{xxx} (overscore over pin or signal name)	Note:	Footnote for item marked with Note in the text	Caution:	Information requiring particular attention	Remark:	Supplementary information	Numeral representation:	Binaryxxxx or xxxxB		Decimalxxxx		HexadecimalxxxxH
Data significance:	Higher digits on the left and lower digits on the right																
Active low representation:	\overline{xxx} (overscore over pin or signal name)																
Note:	Footnote for item marked with Note in the text																
Caution:	Information requiring particular attention																
Remark:	Supplementary information																
Numeral representation:	Binaryxxxx or xxxxB																
	Decimalxxxx																
	HexadecimalxxxxH																

Related Documents

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

Device-related documents

Document Name	Document Number
78K0/KB1+ User's Manual	U16846E
78K0/KC1+ User's Manual	U16961E
78K0/KD1+ User's Manual	U16962E
78K0/KE1+ User's Manual	U16899E
78K0/KF1+ User's Manual	U16819E
78K/0 Series Instructions User's Manual	U12326E

CONTENTS

CHAPTER 1 FLASH MEMORY PROGRAMMING	14
1.1 Overview	14
1.2 System Configuration.....	15
1.3 Programming Overview.....	16
1.3.1 Setting flash memory programming mode	16
1.3.2 Selecting serial communication mode	16
1.3.3 Manipulating flash memory via command transmission/reception	17
1.4 Information Specific to 78K0/Kx1+	17
CHAPTER 2 PROGRAMMER OPERATING ENVIRONMENT.....	19
2.1 Programmer Control Pins	19
2.2 Details of control pins	20
2.2.1 Flash memory programming mode setting pins (FLMD0, FLMD1)	20
2.2.2 Serial interface pins (SI, SO, \overline{SCK} , TxD, RxD, HS)	21
2.2.3 Reset control pin (\overline{RESET})	22
2.2.4 Clock control pin (CLK).....	22
2.2.5 V _{DD} /GND control pins	23
2.2.6 Other pins.....	23
2.3 Basic Flowchart	24
2.4 Setting Flash Memory Programming Mode.....	25
2.5 Selecting Serial Communication Mode	26
2.6 UART Communication Mode	26
2.7 3-Wire Serial I/O Communication Mode with Handshake Supported (CSI + HS).....	27
2.8 3-Wire Serial I/O Communication Mode (CSI)	27
2.9 Shutting Down Target Power Supply.....	27
2.10 Manipulation of Flash Memory	28
2.11 Command List.....	28
2.12 Status List.....	29
CHAPTER 3 BASIC PROGRAMMER OPERATION	30
CHAPTER 4 COMMAND/DATA FRAME FORMAT	31
4.1 Command Frame Transmission Processing.....	33
4.2 Data Frame Transmission Processing	33
4.3 Data Frame Reception Processing	33
CHAPTER 5 DESCRIPTION OF COMMAND PROCESSING.....	34
5.1 Status Command	34
5.1.1 Description.....	34
5.1.2 Command frame and status frame	34
5.2 Reset Command.....	35
5.2.1 Description.....	35
5.2.2 Command frame and status frame	35

5.3	Baud Rate Set Command	36
5.3.1	Description	36
5.3.2	Command frame and status frame.....	36
5.4	Oscillating Frequency Set Command	37
5.4.1	Description	37
5.4.2	Command frame and status frame.....	37
5.5	Chip Erase Command	38
5.5.1	Description	38
5.5.2	Command frame and status frame.....	38
5.6	Block Erase Command	39
5.6.1	Description	39
5.6.2	Command frame and status frame.....	39
5.7	Programming Command	40
5.7.1	Description	40
5.7.2	Command frame and status frame.....	40
5.7.3	Data frame and status frame	40
5.7.4	Completion of transferring all data and status frame.....	41
5.8	Verify Command	42
5.8.1	Description	42
5.8.2	Command frame and status frame.....	42
5.8.3	Data frame and status frame	42
5.9	Block Blank Check Command	44
5.9.1	Description	44
5.9.2	Command frame and status frame.....	44
5.10	Silicon Signature Command	45
5.10.1	Description	45
5.10.2	Command frame and status frame.....	45
5.10.3	Silicon signature data frame	45
5.11	Version Get Command	47
5.11.1	Description	47
5.11.2	Command frame and status frame.....	47
5.11.3	Version data frame.....	48
5.12	Checksum Command	49
5.12.1	Description	49
5.12.2	Command frame and status frame.....	49
5.12.3	Checksum data frame.....	49
5.13	Security Set Command	50
5.13.1	Description	50
5.13.2	Command frame and status frame.....	50
5.13.3	Data frame and status frame	50
5.13.4	Internal verify check and status frame	51
CHAPTER 6	UART COMMUNICATION MODE	53
6.1	Command Frame Transmission Processing Flowchart	53
6.2	Data Frame Transmission Processing Flowchart	54
6.3	Data Frame Reception Processing Flowchart	55
6.4	Reset Command	56
6.4.1	Processing sequence chart.....	56

6.4.2	Description of processing sequence	57
6.4.3	Status at processing completion	57
6.4.4	Flowchart.....	58
6.4.5	Sample program	59
6.5	Baud Rate Set Command	60
6.5.1	Processing sequence chart	60
6.5.2	Description of processing sequence	61
6.5.3	Status at processing completion	61
6.5.4	Flowchart.....	62
6.5.5	Sample program	63
6.6	Oscillating Frequency Set Command	65
6.6.1	Processing sequence chart	65
6.6.2	Description of processing sequence	66
6.6.3	Status at processing completion	66
6.6.4	Flowchart.....	67
6.6.5	Sample program	68
6.7	Chip Erase Command.....	69
6.7.1	Processing sequence chart	69
6.7.2	Description of processing sequence	70
6.7.3	Status at processing completion	70
6.7.4	Flowchart.....	71
6.7.5	Sample program	72
6.8	Block Erase Command.....	73
6.8.1	Processing sequence chart	73
6.8.2	Description of processing sequence	74
6.8.3	Status at processing completion	74
6.8.4	Flowchart.....	75
6.8.5	Sample program	76
6.9	Programming Command	77
6.9.1	Processing sequence chart	77
6.9.2	Description of processing sequence	78
6.9.3	Status at processing completion	79
6.9.4	Flowchart.....	80
6.9.5	Sample program	81
6.10	Verify Command	83
6.10.1	Processing sequence chart	83
6.10.2	Description of processing sequence	84
6.10.3	Status at processing completion	84
6.10.4	Flowchart.....	85
6.10.5	Sample program	86
6.11	Block Blank Check Command.....	88
6.11.1	Processing sequence chart	88
6.11.2	Description of processing sequence	89
6.11.3	Status at processing completion	89
6.11.4	Flowchart.....	90
6.11.5	Sample program	91
6.12	Silicon Signature Command.....	92
6.12.1	Processing sequence chart	92

6.12.2	Description of processing sequence	93
6.12.3	Status at processing completion	93
6.12.4	Flowchart	94
6.12.5	Sample program	95
6.13	Version Get Command	96
6.13.1	Processing sequence chart.....	96
6.13.2	Description of processing sequence	97
6.13.3	Status at processing completion	97
6.13.4	Flowchart	98
6.13.5	Sample program	99
6.14	Checksum Command	100
6.14.1	Processing sequence chart.....	100
6.14.2	Description of processing sequence	101
6.14.3	Status at processing completion	101
6.14.4	Flowchart	102
6.14.5	Sample program	103
6.15	Security Set Command.....	104
6.15.1	Processing sequence chart.....	104
6.15.2	Description of processing sequence	105
6.15.3	Status at processing completion	105
6.15.4	Flowchart	106
6.15.5	Sample program	107

CHAPTER 7 3-WIRE SERIAL I/O COMMUNICATION MODE WITH HANDSHAKE SUPPORTED (CSI + HS)..... 109

7.1	Command Frame Transmission Processing Flowchart.....	109
7.2	Data Frame Transmission Processing Flowchart	110
7.3	Data Frame Reception Processing Flowchart.....	111
7.4	Status Command.....	112
7.4.1	Processing sequence chart.....	112
7.4.2	Description of processing sequence	113
7.4.3	Status at processing completion	113
7.4.4	Flowchart	114
7.4.5	Sample program	115
7.5	Reset Command.....	116
7.5.1	Processing sequence chart.....	116
7.5.2	Description of processing sequence	117
7.5.3	Status at processing completion	117
7.5.4	Flowchart	118
7.5.5	Sample program	119
7.6	Oscillating Frequency Set Command	120
7.6.1	Processing sequence chart.....	120
7.6.2	Description of processing sequence	121
7.6.3	Status at processing completion	121
7.6.4	Flowchart	122
7.6.5	Sample program	123
7.7	Chip Erase Command.....	124
7.7.1	Processing sequence chart.....	124

7.7.2	Description of processing sequence	125
7.7.3	Status at processing completion	125
7.7.4	Flowchart.....	126
7.7.5	Sample program	127
7.8	Block Erase Command.....	128
7.8.1	Processing sequence chart	128
7.8.2	Description of processing sequence	129
7.8.3	Status at processing completion	129
7.8.4	Flowchart.....	130
7.8.5	Sample program	131
7.9	Programming Command.....	132
7.9.1	Processing sequence chart	132
7.9.2	Description of processing sequence	133
7.9.3	Status at processing completion	134
7.9.4	Flowchart.....	135
7.9.5	Sample program	136
7.10	Verify Command	138
7.10.1	Processing sequence chart	138
7.10.2	Description of processing sequence	139
7.10.3	Status at processing completion	139
7.10.4	Flowchart.....	140
7.10.5	Sample program	141
7.11	Block Blank Check Command.....	143
7.11.1	Processing sequence chart	143
7.11.2	Description of processing sequence	144
7.11.3	Status at processing completion	144
7.11.4	Flowchart.....	145
7.11.5	Sample program	146
7.12	Silicon Signature Command.....	147
7.12.1	Processing sequence chart	147
7.12.2	Description of processing sequence	148
7.12.3	Status at processing completion	148
7.12.4	Flowchart.....	149
7.12.5	Sample program	150
7.13	Version Get Command	151
7.13.1	Processing sequence chart	151
7.13.2	Description of processing sequence	152
7.13.3	Status at processing completion	152
7.13.4	Flowchart.....	153
7.13.5	Sample program	154
7.14	Checksum Command	155
7.14.1	Processing sequence chart	155
7.14.2	Description of processing sequence	156
7.14.3	Status at processing completion	156
7.14.4	Flowchart.....	157
7.14.5	Sample program	158
7.15	Security Set Command.....	159
7.15.1	Processing sequence chart	159

7.15.2	Description of processing sequence	160
7.15.3	Status at processing completion	161
7.15.4	Flowchart	162
7.15.5	Sample program	163
CHAPTER 8	3-WIRE SERIAL I/O COMMUNICATION MODE (CSI)	165
8.1	Command Frame Transmission Processing Flowchart.....	165
8.2	Data Frame Transmission Processing Flowchart	166
8.3	Data Frame Reception Processing Flowchart.....	167
8.4	Status Command.....	168
8.4.1	Processing sequence chart.....	168
8.4.2	Description of processing sequence	169
8.4.3	Status at processing completion	169
8.4.4	Flowchart	170
8.4.5	Sample program	171
8.5	Reset Command	172
8.5.1	Processing sequence chart.....	172
8.5.2	Description of processing sequence	173
8.5.3	Status at processing completion	173
8.5.4	Flowchart	174
8.5.5	Sample program	175
8.6	Oscillating Frequency Set Command	176
8.6.1	Processing sequence chart.....	176
8.6.2	Description of processing sequence	177
8.6.3	Status at processing completion	177
8.6.4	Flowchart	178
8.6.5	Sample program	179
8.7	Chip Erase Command.....	180
8.7.1	Processing sequence chart.....	180
8.7.2	Description of processing sequence	181
8.7.3	Status at processing completion	181
8.7.4	Flowchart	182
8.7.5	Sample program	183
8.8	Block Erase Command	184
8.8.1	Processing sequence chart.....	184
8.8.2	Description of processing sequence	185
8.8.3	Status at processing completion	185
8.8.4	Flowchart	186
8.8.5	Sample program	187
8.9	Programming Command	188
8.9.1	Processing sequence chart.....	188
8.9.2	Description of processing sequence	189
8.9.3	Status at processing completion	190
8.9.4	Flowchart	191
8.9.5	Sample program	192
8.10	Verify Command.....	194
8.10.1	Processing sequence chart.....	194
8.10.2	Description of processing sequence	195

8.10.3	Status at processing completion	195
8.10.4	Flowchart.....	196
8.10.5	Sample program	197
8.11	Block Blank Check Command.....	199
8.11.1	Processing sequence chart	199
8.11.2	Description of processing sequence	200
8.11.3	Status at processing completion.....	200
8.11.4	Flowchart.....	201
8.11.5	Sample program	202
8.12	Silicon Signature Command.....	203
8.12.1	Processing sequence chart	203
8.12.2	Description of processing sequence	204
8.12.3	Status at processing completion.....	204
8.12.4	Flowchart.....	205
8.12.5	Sample program	206
8.13	Version Get Command	207
8.13.1	Processing sequence chart	207
8.13.2	Description of processing sequence	208
8.13.3	Status at processing completion.....	208
8.13.4	Flowchart.....	209
8.13.5	Sample program	210
8.14	Checksum Command	211
8.14.1	Processing sequence chart	211
8.14.2	Description of processing sequence	212
8.14.3	Status at processing completion.....	212
8.14.4	Flowchart.....	213
8.14.5	Sample program	214
8.15	Security Set Command.....	216
8.15.1	Processing sequence chart	216
8.15.2	Description of processing sequence	217
8.15.3	Status at processing completion.....	217
8.15.4	Flowchart.....	218
8.15.5	Sample program	219
CHAPTER 9	FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS	221
9.1	Flash Memory Programming Mode Setting Time	221
9.2	Programming Characteristics	222
9.3	UART Communication Mode	224
9.4	3-Wire Serial I/O Communication Mode.....	227
CHAPTER 10	ELECTRICAL SPECIFICATIONS (REFERENCE).....	231
APPENDIX A	CIRCUIT DIAGRAM (REFERENCE).....	238

CHAPTER 1 FLASH MEMORY PROGRAMMING

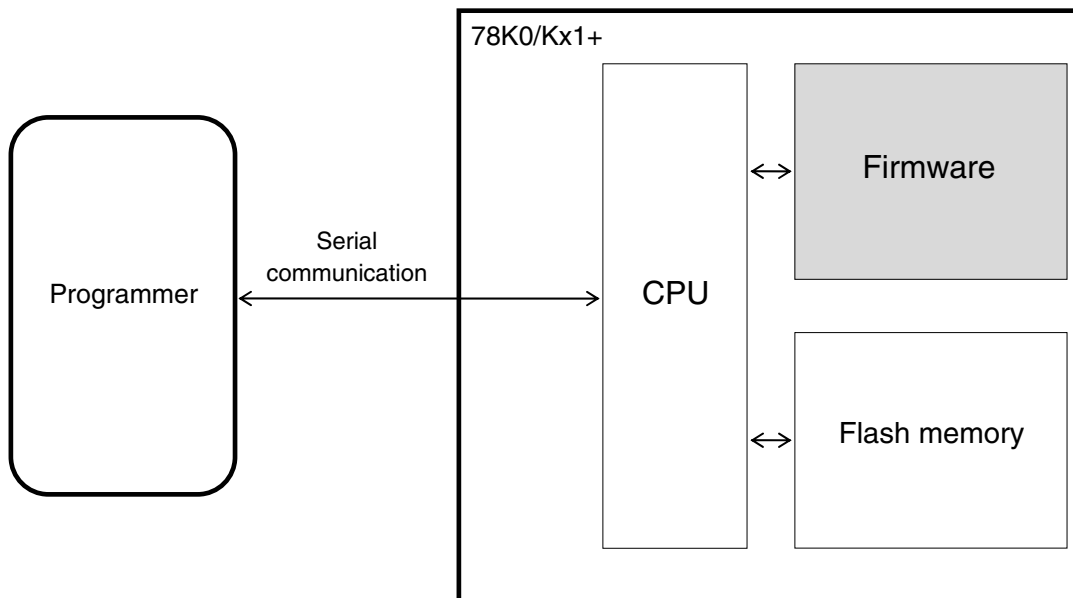
To rewrite the contents of the internal flash memory of the 78K0/Kx1+, a dedicated flash memory programmer (hereafter referred to as the “programmer”) is usually used.

This Application Note explains how to develop a dedicated programmer.

1.1 Overview

The 78K0/Kx1+ incorporates firmware that controls flash memory programming. The programming to the internal flash memory is performed by transmitting/receiving commands between the programmer and the 78K0/Kx1+ via serial communication.

Figure 1-1. System Outline of Flash Memory Programming in 78K0/Kx1+



1.2 System Configuration

Examples of the system configuration for programming the flash memory are illustrated in Figure 1-2.

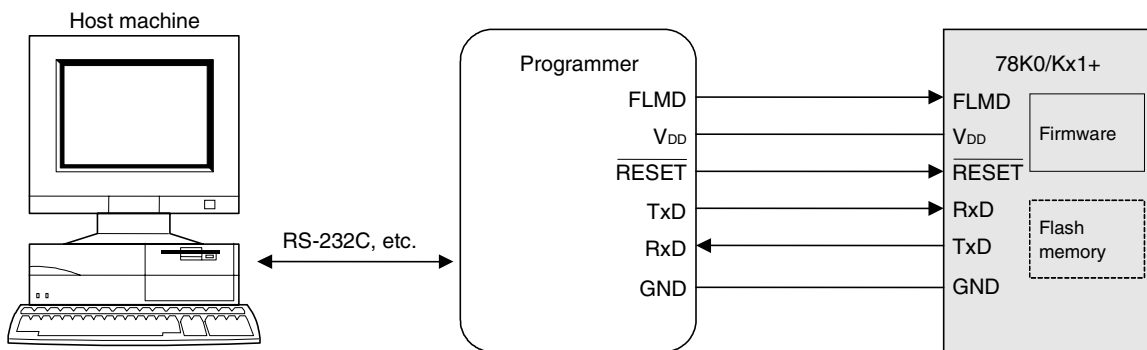
These figures illustrate how to program the flash memory with the programmer, under control of a host machine.

Depending on how the programmer is connected, the programmer can be used in a standalone mode without using the host machine, if a user program has been downloaded to the programmer in advance.

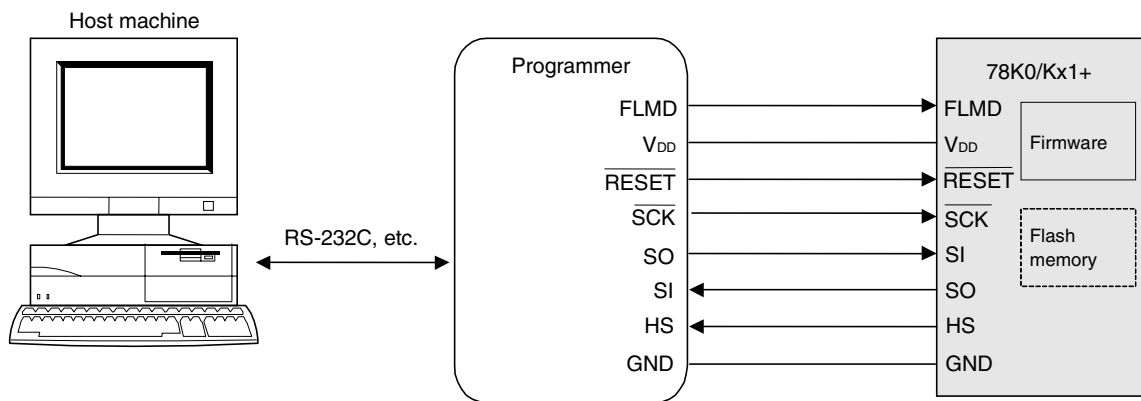
For example, NEC Electronics' flash memory programmer PG-FP4 can execute programming either by using the GUI software with a host machine connected or by itself (standalone).

Figure 1-2. System Configuration Examples

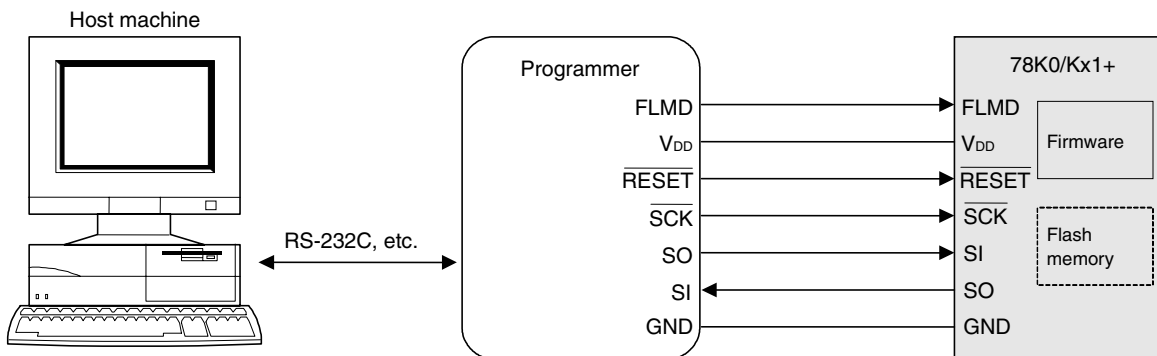
(1) UART communication mode (LSB-first transfer)



(2) 3-wire serial I/O communication mode with handshake supported (CSI + HS) (MSB-first transfer)



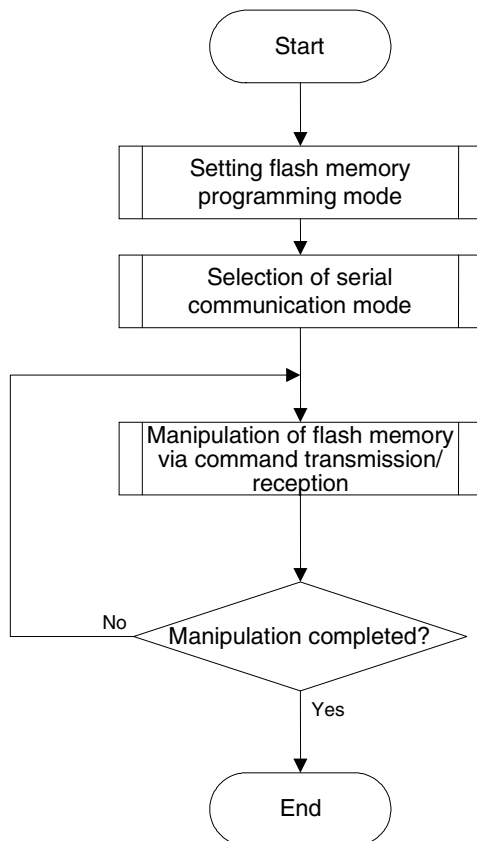
(3) 3-wire serial I/O communication mode (CSI) (MSB-first transfer)



1.3 Programming Overview

To rewrite the contents of the flash memory with the programmer, the 78K0/Kx1+ must first be set to the flash memory programming mode. After that, select the mode for communication between the programmer and the 78K0/Kx1+, transmit commands from the programmer via serial communication, and then rewrite the flash memory. The flowchart of programming is illustrated in Figure 1-3.

Figure 1-3. Programming Flowchart



1.3.1 Setting flash memory programming mode

Supply a specific voltage to the flash memory programming mode setting pins (FLMD0 and FLMD1) in the 78K0/Kx1+ and release a reset; the flash memory programming mode is then set.

1.3.2 Selecting serial communication mode

To select a serial communication mode, generate pulses by changing the voltage at flash memory programming mode setting pin (FLMD0) between the V_{DD} voltage and GND voltage in the flash memory programming mode, and determine the communication mode according to the pulse count.

1.3.3 Manipulating flash memory via command transmission/reception

The flash memory incorporated in the 78K0/Kx1+ has functions to rewrite the flash memory contents. The flash memory manipulating functions shown in Table 1-1 are available.

Table 1-1. Outline of Flash Memory Functions

Function	Outline
Erase	Erases the flash memory contents.
Write	Writes data to the flash memory.
Verify	Compares the flash memory contents with data for verify.
Acquisition of information	Reads information related to the flash memory.

To control these functions, the programmer transmits commands to the 78K0/Kx1+ via serial communication. The 78K0/Kx1+ returns the response status for the commands. The programming to the flash memory is performed by repeating these series of serial communications.

1.4 Information Specific to 78K0/Kx1+

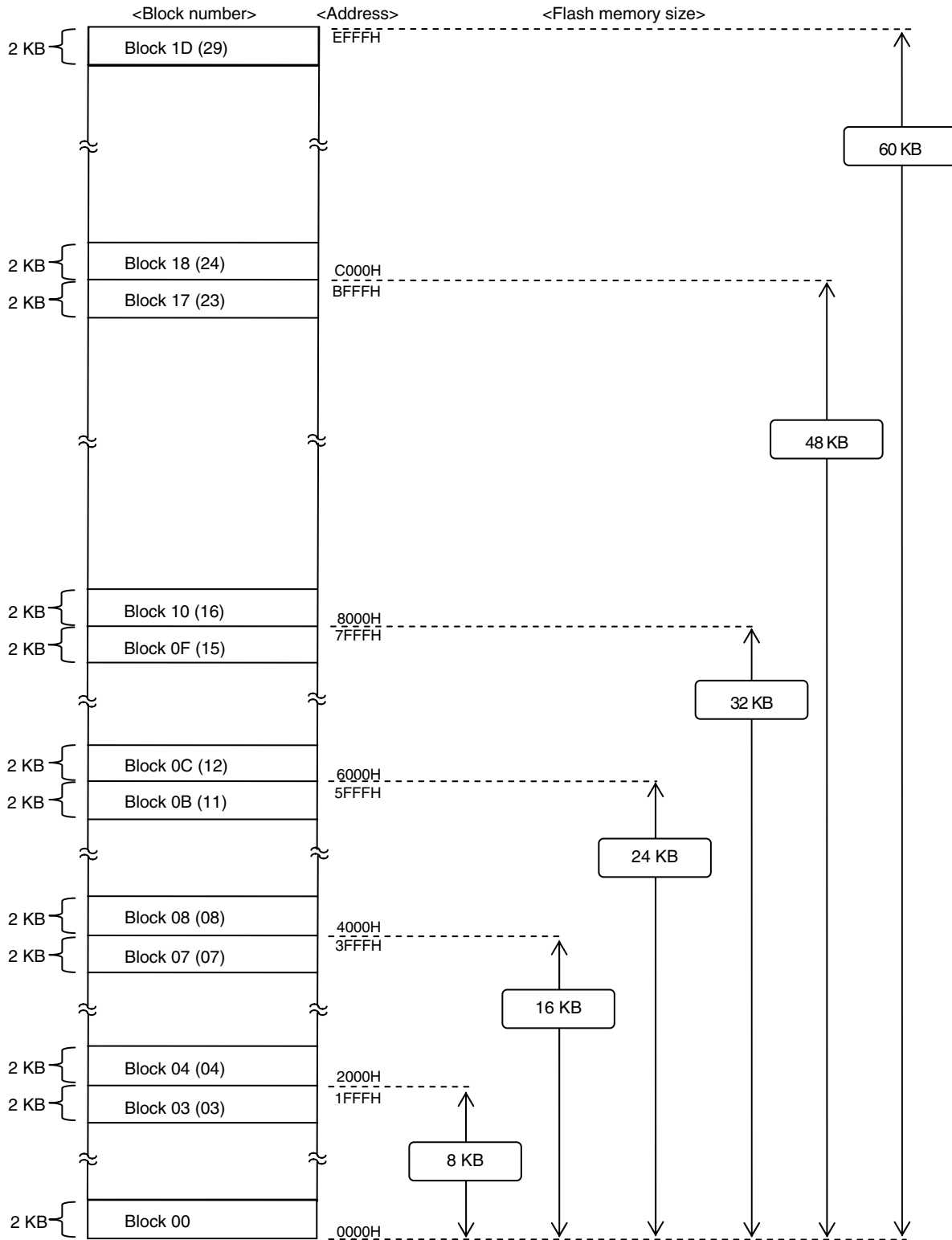
The programmer must manage product-specific information (such as a device name and memory information).

Table 1-2 shows the flash memory size of the 78K0/Kx1+ and Figure 1-4 shows the configuration of the flash memory.

Table 1-2. Flash Memory Size of 78K0/Kx1+

Device Name	Flash Memory Size
78K0/KB1+	μ PD78F0101H 8 KB
	μ PD78F0102H 16 KB
	μ PD78F0103H 24 KB
78K0/KC1+	μ PD78F0112H 16 KB
	μ PD78F0113H 24 KB
	μ PD78F0114H 32 KB
	μ PD78F0114HD 32 KB
78K0/KD1+	μ PD78F0122H 16 KB
	μ PD78F0123H 24 KB
	μ PD78F0124H 32 KB
	μ PD78F0124HD 32 KB
78K0/KE1+	μ PD78F0132H 16 KB
	μ PD78F0133H 24 KB
	μ PD78F0134H 32 KB
	μ PD78F0136H 48 KB
	μ PD78F0138H 60 KB
	μ PD78F0138HD 60 KB
78K0/KF1+	μ PD78F0148H 60 KB
	μ PD78F0148HD 60 KB

Figure 1-4. Flash Memory Configuration



Remark Each block consists of 2 KB (this figure only illustrates some parts of entire blocks in the flash memory).

CHAPTER 2 PROGRAMMER OPERATING ENVIRONMENT

2.1 Programmer Control Pins

Table 2-1 lists the pins that the programmer must control to implement the programmer function in the user system. See the following pages for details on each pin.

Table 2-1. Pin Description

Programmer			78K0/Kx1+	Mode for Communication with Target		
Signal Name	I/O	Pin Function	Pin Name	CSI	CSI + HS	UART
FLMD0	Output	Output of signal level to set programming mode and output of pulse to select communication mode	FLMD0	○	○	○
FLMD1	Output	Output of signal level to set programming mode	FLMD1	○	○	○
V _{DD}	Output	V _{DD} voltage generation/monitoring	V _{DD} EV _{DD} AV _{REF}	△	△	△
GND	–	Ground	V _{SS} EV _{SS} AV _{SS}	○	○	○
CLK	Output	Operating clock output to 78K0/Kx1+	X1, X2 ^{Note}	△	△	△
$\overline{\text{RESET}}$	Output	Programming mode switching trigger	$\overline{\text{RESET}}$	○	○	○
SO	Output	Command transmission to 78K0/Kx1+	SI10	○	○	×
SI	Input	Response status and data reception from 78K0/Kx1+	SO10	○	○	×
$\overline{\text{SCK}}$	Output	Serial clock supply to 78K0/Kx1+	$\overline{\text{SCK}}10$	○	○	×
HS (handshake)	Input	Handshake signal reception for serial communication with 78K0/Kx1+	HS	×	○	×
TxD	Output	Command transmission to 78K0/Kx1+	RxD6	×	×	○
RxD	Input	Response status and data reception from 78K0/Kx1+	TxD6	×	×	○

Note Connect the X1 inverse signal to X2.

Remark ○: Be sure to connect the pin.

×: The pin does not have to be connected.

△: The pin does not have to be connected if the signal is generated on the target board.

For the voltage of the pins controlled by the programmer, refer to the user's manual of the device that is subject to flash memory programming.

2.2 Details of control pins

2.2.1 Flash memory programming mode setting pins (FLMD0, FLMD1)

The FLMD0 and FLMD1 pins are used to control the operating mode of the 78K0/Kx1+. The 78K0/Kx1+ operates in flash memory programming mode when a specific voltage is supplied to these pins and a reset is released.

The mode for the serial communication between the programmer and the 78K0/Kx1+ is determined by controlling the voltage at the FLMD0 pin between V_{DD} and GND and outputting pulses, after reset. Refer to Table 2-3 for the relationship between the FLMD0 pulse counts and communication modes.

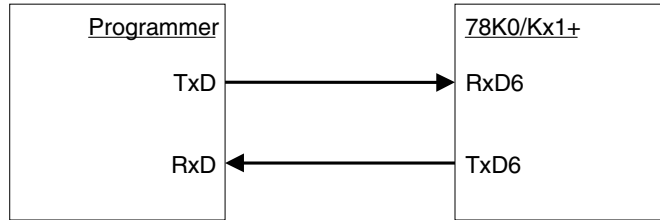
2.2.2 Serial interface pins (SI, SO, SCK, TxD, RxD, HS)

The serial interface pins are used to transfer the flash memory writing commands between the programmer and the 78K0/Kx1+.

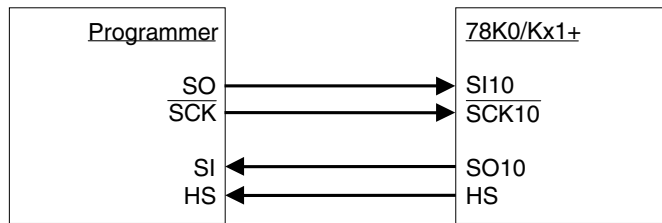
With the 78K0/Kx1+, the communication mode can be selected from UART, CSI + HS, and CSI. The following figures illustrate the connection of pins used in each communication mode.

Figure 2-1. Serial Interface Pins

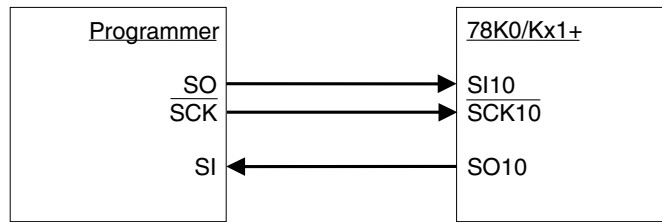
(1) UART communication mode



(2) 3-wire serial I/O communication mode with handshake supported (CSI + HS)



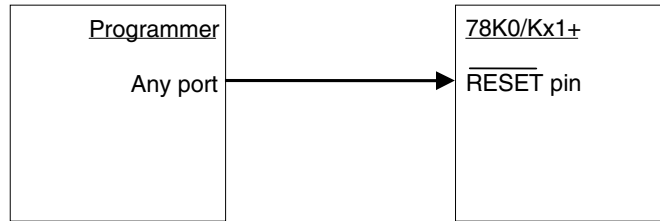
(3) 3-wire serial I/O communication mode (CSI)



2.2.3 Reset control pin ($\overline{\text{RESET}}$)

The reset control pin is used to control the system reset for the 78K0/Kx1+ from the programmer. The flash memory programming mode can be selected when a specific voltage is supplied to the FLMD0 and FLMD1 pins and a reset is released.

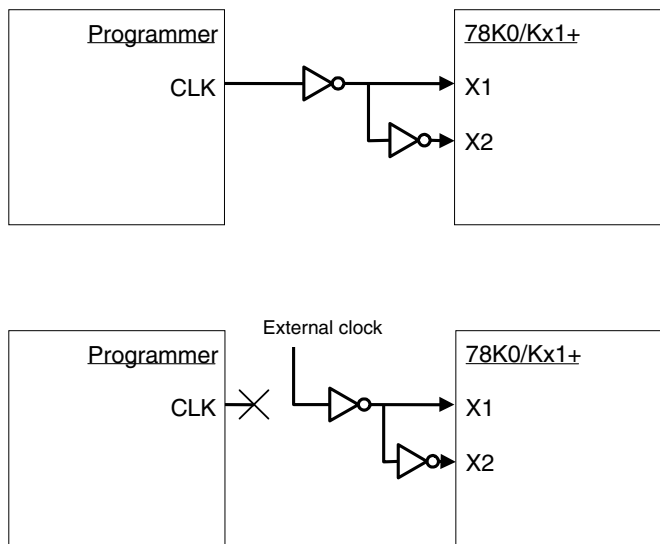
Figure 2-2. Reset Control Pin



2.2.4 Clock control pin (CLK)

The clock control pin is used only when the clock is supplied from the programmer to the 78K0/Kx1+. Connection of this pin is not necessary when it is not necessary to supply the operating clock to the 78K0/Kx1+ from the programmer.

Figure 2-3. Clock Control Pin

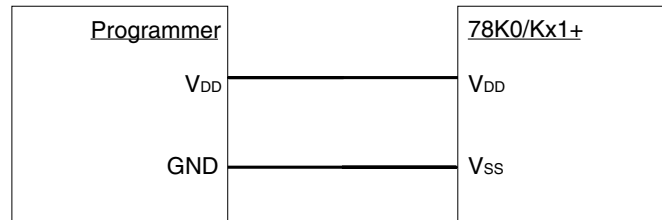


2.2.5 V_{DD} /GND control pins

The V_{DD} control pin is used to supply power to the 78K0/Kx1+ from the programmer. Connection of this pin is not necessary when it is not necessary to supply power to the 78K0/Kx1+ from the programmer. However, this pin must be connected regardless of whether the power is supplied from the programmer when the dedicated programmer is used, because the dedicated programmer monitors the power supply status of the 78K0/Kx1+.

The GND control pin must be connected to V_{SS} of the 78K0/Kx1+ regardless of whether the power is supplied from the programmer.

Figure 2-4. V_{DD} /GND Control Pin



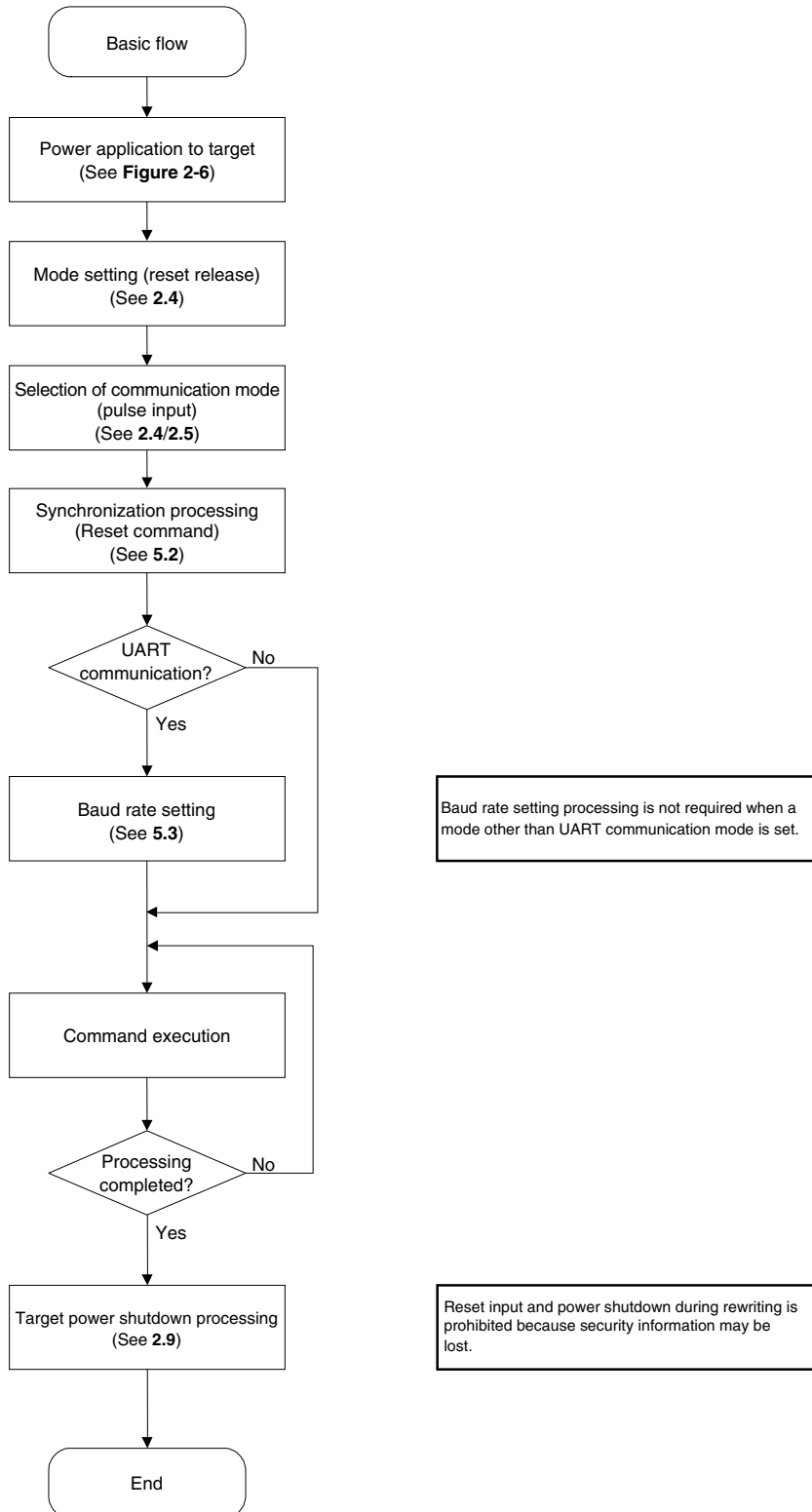
2.2.6 Other pins

For the connection of the pins that are not connected to the programmer, refer to the chapter describing the flash memory in the user's manual of each device.

2.3 Basic Flowchart

The following illustrates the basic flowchart for performing flash memory rewriting with the programmer.

Figure 2-5. Basic Flowchart for Flash Memory Rewrite Processing

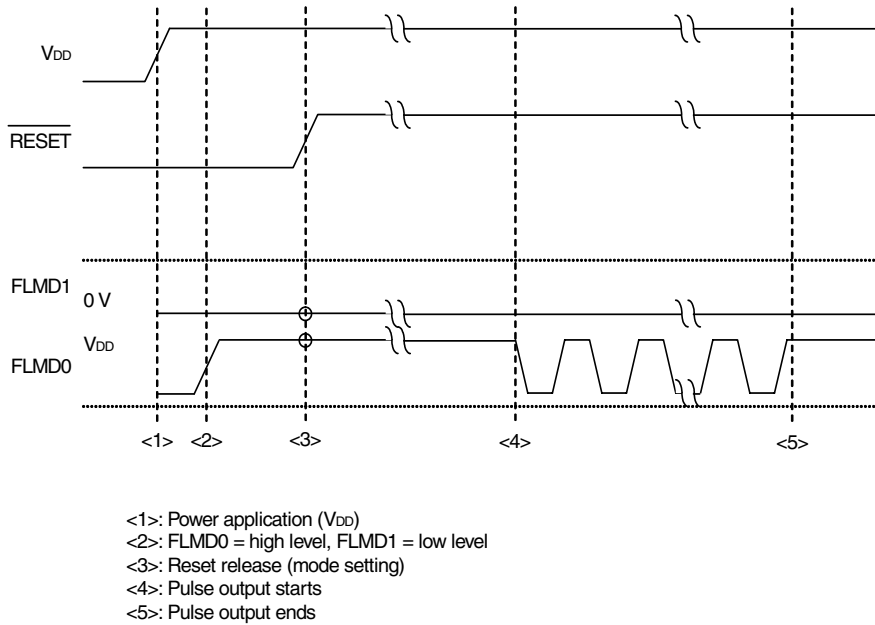


2.4 Setting Flash Memory Programming Mode

To rewrite the contents of the flash memory with the programmer, the 78K0/Kx1+ must first be set to the flash memory programming mode by supplying a specific voltage to the flash memory programming mode setting pins (FLMD0, FLMD1) in the 78K0/Kx1+, then releasing a reset.

The following illustrates a timing chart for setting the flash memory programming mode and selecting the communication mode.

Figure 2-6. Setting Flash Memory Programming Mode and Selecting Communication Mode



The relationship between the settings of the FLMD0 and FLMD1 pins after reset release and the operating mode is shown below.

Table 2-2. Relationship Between Settings of FLMD0 and FLMD1 Pins After Reset Release and Operating Mode

FLMD0	FLMD1	Operating Mode
Low (GND)	Any	Normal operating mode
High (V_{DD})	Low (GND)	Flash memory programming mode
High (V_{DD})	High (V_{DD})	Setting prohibited

2.5 Selecting Serial Communication Mode

The communication mode is determined by inputting a pulse to the FLMD0 pin in the 78K0/Kx1+ after reset release. The high- and low-levels of the FLMD0 pulse are V_{DD} and GND, respectively.

The following table shows the relationship between the number of FLMD0 pulses (pulse counts) and communication modes that can be selected with the 78K0/Kx1+.

Table 2-3. Relationship Between FLMD0 Pluse Counts and Communication Modes

Communication Mode	FLMD0 Pulse Counts	Port Used for Communication
UART (UART6)	0	TxD6 (P13), RxD6 (P14)
3-wire serial I/O (CSI10)	8	SO10 (P12), SI10 (P11), $\overline{SCK10}$ (P10)
3-wire serial I/O with handshake supported (CSI10 + HS)	11	SO10 (P12), SI10 (P11), $\overline{SCK10}$ (P10), HS (P15)
Setting prohibited	Others	—

2.6 UART Communication Mode

The RxD and TxD pins are used for UART communication. The communication conditions are as shown below.

Table 2-4. UART Communication Conditions

Item	Description
Baud rate	Selectable from 9,600, 19,200, 31,250, 38,400, 76,800, and 153,600 bps (default: 9,600 bps)
Parity bit	None
Data length	8 bits (LSB first)
Stop bit	1 bit

The programmer always operates as the master device during CSI communication, so the programmer must check whether the processing by the 78K0/Kx1+, such as writing or erasing, is normally completed. On the other hand, the status of the master and slave is occasionally exchanged during UART communication, so communication at the optimum timing is possible without assigning one pin like CSI + HS communication.

Caution Set the same baud rate to the master and slave devices when performing UART communication.

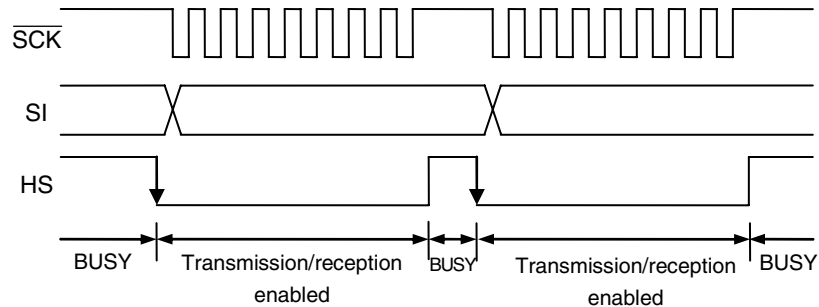
2.7 3-Wire Serial I/O Communication Mode with Handshake Supported (CSI + HS)

In the CSI + HS communication mode, the timing for communication of commands or data is optimized. In addition to the SI, SO and $\overline{\text{SCK}}$ pins, the HS (handshake) pin is used for implementing effective communication.

The level of the HS pin signal falls (low level) when the 78K0/Kx1+ is ready for transmitting or receiving data. The programmer must check the falling edge of the HS pin signal (low level) before starting transmission/reception of commands or data to the 78K0/Kx1+.

The communication data format is MSB-first, in 8-bit units. Keep the clock frequency 2.5 MHz or lower.

Figure 2-7. Timing Chart of CSI + HS Communication



2.8 3-Wire Serial I/O Communication Mode (CSI)

The $\overline{\text{SCK}}$, SO and SI pins are used for CSI communication. The programmer always operates as the master device, so communication may not be performed normally if data is transmitted via the $\overline{\text{SCK}}$ pin while the 78K0/Kx1+ is not ready for transmission/reception.

The communication data format is MSB-first, in 8-bit units. Keep the clock frequency 2.5 MHz or lower.

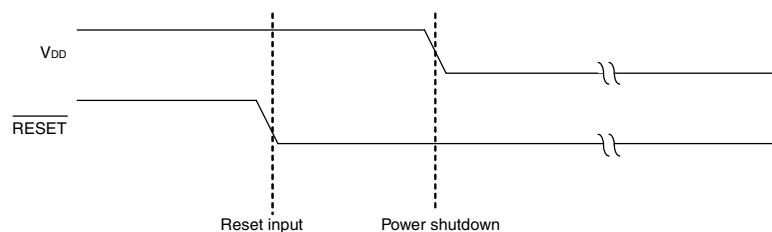
2.9 Shutting Down Target Power Supply

After each command execution is completed, shut down the power supply to the target after setting the $\overline{\text{RESET}}$ pin to low level, as shown below.

Set other pins to Hi-Z when shutting down the power supply to the target.

Caution Shutting down the power supply and inputting a reset during command processing are prohibited.

Figure 2-8. Timing for Terminating Flash Memory Programming Mode



2.10 Manipulation of Flash Memory

The flash memory incorporated in the 78K0/Kx1+ has functions to manipulate the flash memory, as listed in Table 2-5. The programmer transmits commands to control these functions to the 78K0/Kx1+, and checks the response status sent from the 78K0/Kx1+, to manipulate the flash memory.

Table 2-5. List of Flash Memory Manipulating Functions

Classification	Function Name	Description
Erase	Chip erase	Erases the entire flash memory area. Clears the security flag.
	Block erase	Erases a specified block in the flash memory.
Write	Write	Writes data to a specified area in the flash memory.
Verify	Verify	Compares data acquired from a specified address in the flash memory with data transmitted from the programmer, on the 78K0/Kx1+ side.
Blank check	Block blank check	Checks the erase status of a specified area in the flash memory.
Information acquisition	Silicon signature acquisition	Acquires writing protocol information.
	Version acquisition	Acquires version information of the 78K0/Kx1+ and firmware.
	Status acquisition	Acquires the current operating status.
	Checksum acquisition	Acquires checksum data of a specified area.
Security	Security setting	Sets security information.
Other	Reset	Detects synchronization in communication.

2.11 Command List

The commands used by the programmer and their functions are listed below.

Table 2-6. List of Commands Transmitted from Programmer to 78K0/Kx1+

Command Number	Command Name	Function
70H	Status	Acquires the current operating status (status data).
00H	Reset	Detects synchronization in communication.
90H	Oscillating Frequency Set	Specifies the oscillation frequency of the 78K0/Kx1+.
9AH	Baud Rate Set	Sets baud rate when UART communication mode is selected.
20H	Chip Erase	Erases the entire flash memory area.
22H	Block Erase	Erases a specified area in the flash memory.
40H	Programming	Writes data to a specified area in the flash memory.
13H	Verify	Compares the contents in a specified area in the flash memory with data transmitted from the programmer.
32H	Block Blank Check	Checks the erase status of a specified block in the flash memory.
C0H	Silicon Signature	Acquires 78K0/Kx1+ information (part number, flash memory configuration, etc.).
C5H	Version Get	Acquires version information of the 78K0/Kx1+ and firmware.
B0H	Checksum	Acquires checksum data of a specified area.
A0H	Security Set	Sets security information.

2.12 Status List

The following table lists the status codes the programmer receives from the 78K0/Kx1+.

Table 2-7. Status Code List

Status Code	Status	Description
04H	Command number error	Error returned if a command not supported is received
05H	Parameter error	Error returned if command information (parameter) is invalid
06H	Normal acknowledgment (ACK)	Normal acknowledgment
07H	Checksum error	Error returned if data in a frame transmitted from the programmer is abnormal
0FH	Verify error	Error returned if a verify error has occurred upon verifying data transmitted from the programmer
10H	Protect error	Error returned if an attempt is made to execute processing that is prohibited by the Security Set command
15H	Negative acknowledgment (NACK)	Negative acknowledgment
18H	FLMD error	Error returned when a write error has occurred
1AH	MRG10 error	Erase verify error
1BH	MRG11 error	Internal verify error or blank check error during data write
1CH	Write error	Write error
FFH	Processing in progress (BUSY)	Busy response ^{Note}

Note During CSI communication, 1-byte “FFH” may be transmitted, as well as “FFH” as the data frame format.

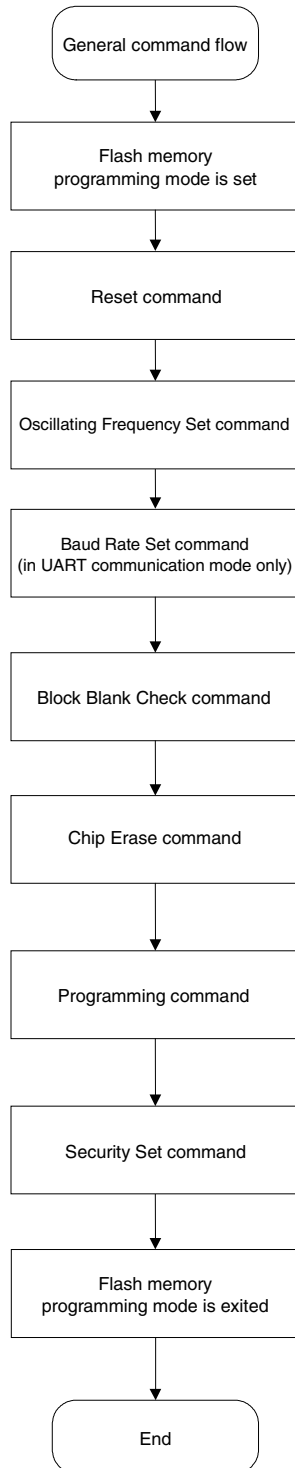
Reception of a checksum error or NACK is treated as an immediate abnormal end in this manual. When a dedicated programmer is developed, however, the processing may be retried without problem from the wait immediately before transmission of the command that results a checksum error or NACK or after BUSY status check via the HS pin. In this event, limiting the retry count is recommended for preventing infinite repetition of the retry operation.

Although not listed in the above table, if a time-out error (BUSY time-out, HS pin time-out, or time-out in data frame reception during UART communication) occurs, it is recommended to shutdown the power supply to the 78K0/Kx1+ (refer to **2.9 Shutting Down Target Power Supply**) and then connect the power supply again.

CHAPTER 3 BASIC PROGRAMMER OPERATION

Figure 3-1 illustrates the general command execution flow when flash memory rewriting is performed with the programmer.

Figure 3-1. General Command Execution Flow at Flash Memory Rewriting



Remark The Verify command and Checksum command can also be supported.

CHAPTER 4 COMMAND/DATA FRAME FORMAT

The programmer uses the command frame to transmit commands to the 78K0/Kx1+. The 78K0/Kx1+ uses the data frame to transmit write data or verify data to the programmer. A header, footer, data length information, and checksum are appended to each frame to enhance the reliability of the transferred data.

The following shows the format of a command frame and data frame.

Figure 4-1. Command Frame Format

SOH (1 byte)	LEN (1 byte)	COM (1 byte)	Command information (variable length) (Max. 255 bytes)	SUM (1 byte)	ETX (1 byte)
-----------------	-----------------	-----------------	---	-----------------	-----------------

Figure 4-2. Data Frame Format

STX (1 byte)	LEN (1 byte)	Data (variable length) (Max. 256 bytes)	SUM (1 byte)	ETX or ETB (1 byte)
-----------------	-----------------	--	-----------------	------------------------

Table 4-1. Description of Symbols in Each Frame

Symbol	Value	Description
SOH	01H	Command frame header
STX	02H	Data frame header
LEN	–	Data length information (00H indicates 256). Command frame: COM + command information length Data frame: Data field length
COM	–	Command number
SUM	–	Checksum data for a frame Obtained by sequentially subtracting all of calculation target data from the initial value (00H) in 1-byte units (borrow is ignored). The calculation targets are as follows. Command frame: LEN + COM + all of command information Data frame: LEN + all of data
ETB	17H	Footer of data frame other than the last frame
ETX	03H	Command frame footer, or footer of last data frame

The following shows examples of calculating the checksum (SUM) for a frame.

[Command frame]

No command information is included in the following example of a Status command frame, so LEN and COM are targets of checksum calculation.

SOH	LEN	COM	SUM	ETX
01H	01H	70H	Checksum	03H
Checksum calculation targets				

For this command frame, checksum data is obtained as follows.

$$00H \text{ (initial value)} - 01H \text{ (LEN)} - 70H \text{ (COM)} = 8FH \text{ (Borrow ignored. Lower 8 bits only.)}$$

The command frame finally transmitted is as follows.

SOH	LEN	COM	SUM	ETX
01H	01H	70H	8FH	03H

[Data frame]

To transmit a data frame as shown below, LEN and D1 to D4 are targets of checksum calculation.

STX	LEN	D1	D2	D3	D4	SUM	ETX
02H	04H	FFH	80H	40H	22H	Checksum	03H
checksum calculation targets							

For this data frame, checksum data is obtained as follows.

$$00H \text{ (initial value)} - 04H \text{ (LEN)} - FFH \text{ (D1)} - 80H \text{ (D2)} - 40H \text{ (D3)} - 22H \text{ (D4)} \\ = 1BH \text{ (Borrow ignored. Lower 8 bits only.)}$$

The data frame finally transmitted is as follows.

STX	LEN	D1	D2	D3	D4	SUM	ETX
02H	04H	FFH	80H	40H	22H	1BH	03H

When a data frame is received, the checksum data is calculated in the same manner, and the obtained value is used to detect a checksum error by judging whether the value is the same as that stored in the SUM field of the receive data. When a data frame as shown below is received, for example, a checksum error is detected.

STX	LEN	D1	D2	D3	D4	SUM	ETX
02H	04H	FFH	80H	40H	22H	1AH	03H

↑ Should be 1BH, if normal

4.1 Command Frame Transmission Processing

Read the following chapters for details on flowcharts of command processing to transmit command frames, for each communication mode.

- For the UART communication mode, read **6.1 Flowchart of Command Frame Transmission Processing**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.1 Flowchart of Command Frame Transmission Processing**.
- For the 3-wire serial I/O communication mode (CSI), read **8.1 Flowchart of Command Frame Transmission Processing**.

4.2 Data Frame Transmission Processing

The write data frame (user program), verify data frame (user program), and security data frame (security flag) are transmitted as a data frame.

Read the following chapters for details on flowcharts of command processing to transmit data frames, for each communication mode.

- For the UART communication mode, read **6.2 Flowchart of Data Frame Transmission Processing**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.2 Flowchart of Data Frame Transmission Processing**.
- For the 3-wire serial I/O communication mode (CSI), read **8.2 Flowchart of Data Frame Transmission Processing**.

4.3 Data Frame Reception Processing

The status frame, silicon signature data frame, version data frame, and checksum data frame are received as a data frame.

Read the following chapters for details on flowcharts of command processing to receive data frames, for each communication mode.

- For the UART communication mode, read **6.3 Flowchart of Data Frame Reception Processing**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.3 Flowchart of Data Frame Reception Processing**.
- For the 3-wire serial I/O communication mode (CSI), read **8.3 Flowchart of Data Frame Reception Processing**.

CHAPTER 5 DESCRIPTION OF COMMAND PROCESSING

5.1 Status Command

5.1.1 Description

This command is used to check the operation status of the 78K0/Kx1+ after issuance of each command such as write or erase.

After the Status command is issued, if the Status command frame cannot be received normally in the 78K0/Kx1+ due to problems based on communication or the like, the status setting will not be performed in the 78K0/Kx1+. As a result, a busy response (FFH), not the status frame, may be received. In such a case, retry the Status command.

5.1.2 Command frame and status frame

Figure 5-1 shows the format of a command frame for the Status command, and Figure 5-2 shows the status frame for the command.

Figure 5-1. Status Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	SUM	ETX
01H	01H	70H (Status)	Checksum	03H

Figure 5-2. Status Frame for Status Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data			SUM	ETX
02H	n	ST1	...	STn	Checksum	03H

- Remarks**
1. ST1 to STn: Status #1 to Status #n
 2. The length of a status frame varies according to each command (such as write or erase) to be transmitted to the 78K0/Kx1+.

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- The Status command is not used in the UART communication mode.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.4 Status Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.4 Status Command**.

Caution After each command such as write or erase is transmitted in UART communication, the 78K0/Kx1+ automatically returns the status frame within a specified time. The Status command is therefore not used.

If the Status command is transmitted in UART communication, the Command Number Error is returned.

5.2 Reset Command

5.2.1 Description

This command is used to check the establishment of communication between the programmer and the 78K0/Kx1+ after the communication mode is set.

When UART is selected as the mode for communication with the 78K0/Kx1+, the same baud rate must be set in the programmer and 78K0/Kx1+. However, the 78K0/Kx1+ cannot detect its own operating frequency so the baud rate cannot be set. It makes detection of the operating frequency in the 78K0/Kx1+ possible by sending "00H" twice at 9,600 bps from the programmer, measuring the low-level width of "00H", and then calculating the average of two sent signals. The baud rate can consequently be set, which enables synchronous detection in communication.

5.2.2 Command frame and status frame

Figure 5-3 shows the format of a command frame for the Reset command, and Figure 5-4 shows the status frame for the command.

Figure 5-3. Reset Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	SUM	ETX
01H	01H	00H (Reset)	Checksum	03H

Figure 5-4. Status Frame for Reset Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	1	ST1	Checksum	03H

Remark ST1: Synchronization detection result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.4 Reset Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.5 Reset Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.5 Reset Command**.

5.3 Baud Rate Set Command

5.3.1 Description

This command is used to change the baud rate for UART (default value: 9,600 bps).

After the Baud Rate Set command is executed, the Reset command must be executed to confirm synchronization at the new baud rate.

The Baud Rate Set command is valid only in the UART communication mode. Data for setting the baud rate is represented as a 1-byte numeric value.

The 78K0/Kx1+ ignores the Baud Rate Set command if it is transmitted in modes other than the UART communication mode.

5.3.2 Command frame and status frame

Figure 5-5 shows the format of a command frame for the Baud Rate Set command, and Figure 5-6 shows the status frame for the command.

Figure 5-5. Baud Rate Set Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information	SUM	ETX
01H	02H	9AH (Baud Rate Set)	D01	Checksum	03H

Remark D01: Baud rate selection value

D01 Value	03H	04H	05H	06H	07H	08H
Baud rate (bps)	9600	19200	31250	38400	76800	153600

Figure 5-6. Status Frame for Baud Rate Set Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Synchronization detection result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.5 Baud Rate Set Command**.
- The Baud Rate Set command is not used in the 3-wire serial I/O communication mode with handshake supported (CSI + HS).
- The Baud Rate Set command is not used in 3-wire serial I/O communication mode (CSI).

5.4 Oscillating Frequency Set Command

5.4.1 Description

This command is used to set oscillation frequency data in the 78K0/Kx1+.

Set the frequency of the clock that is actually input to the X1 pin of the 78K0/Kx1+.

5.4.2 Command frame and status frame

Figure 5-7 shows the format of a command frame for the Oscillating Frequency Set command, and Figure 5-8 shows the status frame for the command.

Figure 5-7. Oscillating Frequency Set Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information				SUM	ETX
01H	05H	90H (Oscillating Frequency Set)	D01	D02	D03	D04	Checksum	03H

Remark D01 to D04: Oscillation frequency = $(D01 \times 0.1 + D02 \times 0.01 + D03 \times 0.001) \times 10^{D04}$ (Unit: kHz)
Settings can be made from 10 kHz to 100 MHz, but set the value according to the specifications of each device when actually transmitting the command.
D01 to D03 hold unpacked BCDs, and D04 holds a signed integer.

Setting example: To set 6 MHz

D01 = 06H

D02 = 00H

D03 = 00H

D04 = 04H

Oscillation frequency = $6 \times 0.1 \times 10^4 = 6,000 \text{ kHz} = 6 \text{ MHz}$

Setting example: To set 10 MHz

D01 = 01H

D02 = 00H

D03 = 00H

D04 = 05H

Oscillation frequency = $1 \times 0.1 \times 10^5 = 10,000 \text{ kHz} = 10 \text{ MHz}$

Figure 5-8. Status Frame for Oscillating Frequency Set Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Oscillation frequency setting result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.6 Oscillating Frequency Set Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.6 Oscillating Frequency Set Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.6 Oscillating Frequency Set Command**.

5.5 Chip Erase Command

5.5.1 Description

This command is used to erase the entire contents of the flash memory. In addition, all of the information that is set by security setting processing can be initialized by chip erase processing, as long as erasure is not prohibited by the security setting (see **5.13 Security Set Command**).

5.5.2 Command frame and status frame

Figure 5-9 shows the format of a command frame for the Chip Erase command, and Figure 5-10 shows the status frame for the command.

Figure 5-9. Chip Erase Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	SUM	ETX
01H	01H	20H (Chip Erase)	Checksum	03H

Figure 5-10. Status Frame for Chip Erase Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Chip erase result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.7 Chip Erase Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.7 Chip Erase Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.7 Chip Erase Command**.

5.6 Block Erase Command

5.6.1 Description

This command is used to erase the contents of blocks with the specified number in the flash memory, as long as erasure is not prohibited by the security setting (see **5.13 Security Set Command**).

5.6.2 Command frame and status frame

Figure 5-11 shows the format of a command frame for the Block Erase command, and Figure 5-12 shows the status frame for the command.

Figure 5-11. Block Erase Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information	SUM	ETX
01H	02H	22H (Block Erase)	BLK	Checksum	03H

Remark BLK: Number of block to be erased

Figure 5-12. Status Frame for Block Erase Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Block erase result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.8 Block Erase Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.8 Block Erase Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.8 Block Erase Command**.

5.7 Programming Command

5.7.1 Description

This command is used to transmit data by the number of written bytes after the write start address and the write end address are transmitted. This command then writes the user program to the flash memory and verifies it internally.

The write start/end address can be set only in the block start/end address units.

If both of the status frames (ST1 and ST2) after the last data transmission indicate ACK, the 78K0/Kx1+ firmware automatically executes internal verify. Therefore, the Status command for this internal verify must be transmitted.

5.7.2 Command frame and status frame

Figure 5-13 shows the format of a command frame for the Programming command, and Figure 5-14 shows the status frame for the command.

Figure 5-13. Programming Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	40H (Programming)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

Remark SAH, SAM, SAL: Write start addresses
EAH, EAM, EAL: Write end addresses

Figure 5-14. Status Frame for Programming Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (a)	Checksum	03H

Remark ST1 (a): Command reception result

5.7.3 Data frame and status frame

Figure 5-15 shows the format of a frame that includes data to be written, and Figure 5-16 shows the status frame for the data.

Figure 5-15. Data Frame to Be Written (from Programmer to 78K0/Kx1+)

STX	LEN	Data	SUM	ETX/ETB
02H	00H to FFH (00H = 256)	Write Data	Checksum	03H/17H

Remark Write Data: User program to be written

Figure 5-16. Status Frame for Data Frame (from 78K0/Kx1+ to Programmer)

STX	LEN	Data		SUM	ETX
02H	02H	ST1 (b)	ST2 (b)	Checksum	03H

Remark ST1 (b): Data reception check result
ST2 (b): Write result

5.7.4 Completion of transferring all data and status frame

Figure 5-17 shows the status frame after transfer of all data is completed.

Figure 5-17. Status Frame After Completion of Transferring All Data (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (c)	Checksum	03H

Remark ST1 (c): Internal verify result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.9 Programming Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.9 Programming Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.9 Programming Command**.

5.8 Verify Command

5.8.1 Description

This command is used to compare the data transmitted from the programmer with the data read from the 78K0/Kx1+ (read level) in the specified address range, and check whether they match.

The verify start/end address can be set only in the block start/end address units.

5.8.2 Command frame and status frame

Figure 5-18 shows the format of a command frame for the Verify command, and Figure 5-19 shows the status frame for the command.

Figure 5-18. Verify Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	13H (Verify)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

Remark SAH, SAM, SAL: Verify start addresses

EAH, EAM, EAL: Verify end addresses

Figure 5-19. Status Frame for Verify Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (a)	Checksum	03H

Remark ST1 (a): Command reception result

5.8.3 Data frame and status frame

Figure 5-20 shows the format of a frame that includes data to be verified, and Figure 5-21 shows the status frame for the data.

Figure 5-20. Data Frame of Data to Be Verified (from Programmer to 78K0/Kx1+)

STX	LEN	Data	SUM	ETX/ETB
02H	00H to FFH (00H = 256)	Verify data	Checksum	03H/17H

Remark Verify Data: User program to be verified

Figure 5-21. Status Frame for Data Frame (from 78K0/Kx1+ to Programmer)

STX	LEN	Data		SUM	ETX
02H	02H	ST1 (b)	ST2 (b)	Checksum	03H

Remark ST1 (b): Data reception check result
 ST2 (b): Verify result^{Note}

Note Even if a verify error occurs in the specified address range, ACK is always returned as the verify result. The status of all verify errors are reflected in the verify result for the last data. Therefore, the occurrence of verify errors can be checked only when all the verify processing for the specified address range is completed.

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.10 Verify Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.10 Verify Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.10 Verify Command**.

5.9 Block Blank Check Command

5.9.1 Description

This command is used to check if a block in the flash memory, with a specified block number, is blank (erased state).

5.9.2 Command frame and status frame

Figure 5-22 shows the format of a command frame for the Block Blank Check command, and Figure 5-23 shows the status frame for the command.

Figure 5-22. Block Blank Check Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information	SUM	ETX
01H	02H	32H (Block Blank Check)	BLK	Checksum	03H

Remark BLK: Number of block to be checked for blank

Figure 5-23. Status Frame for Block Blank Check Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Block blank check result

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.11 Block Blank Check Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.11 Block Blank Check Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.11 Block Blank Check Command**.

5.10 Silicon Signature Command

5.10.1 Description

This command is used to read the write protocol information (silicon signature) of the device.

If the programmer supports a programming protocol that is not supported in the 78K0/Kx1+, for example, execute this command to select an appropriate protocol in accordance with the values of the second and third bytes.

5.10.2 Command frame and status frame

Figure 5-24 shows the format of a command frame for the Silicon Signature command, and Figure 5-25 shows the status frame for the command.

Figure 5-24. Silicon Signature Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	SUM	ETX
01H	01H	C0H (Silicon Signature)	Checksum	03H

Figure 5-25. Status Frame for Silicon Signature Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Command reception result

5.10.3 Silicon signature data frame

Figure 5-26 shows the format of a frame that includes silicon signature data.

Figure 5-26. Silicon Signature Data Frame (from 78K0/Kx1+ to Programmer)

STX	LEN	Data				SUM	ETX
02H	n	VEN	EXT	FNC	Invalid data	Checksum	03H

- Remarks**
1. n (LEN): Data length
 VEN: Vendor code (NEC: 10H)
 EXT: Extension code
 FNC: Function information
 INVALID DATA: Invalid data of 90 to 198-byte length.
 2. For the vendor code (VEN), extension code (EXT) and function information (FNC), the highest bit is used as an odd parity. The following shows an example.

Table 5-1. Example of Silicon Signature Data

Field	Contents	Length (Byte)	Example of Silicon Signature Data ^{Note}	Actual Value
VEN	Vendor code (NEC)	1	10H (00010000B)	10H
EXT	Extension code (fixed)	1	7FH (01111111B)	7FH
FNC	Function information (fixed)	1	01H (00000001B)	01H

Note 0 is an odd parity (the value to adjust the number of "1" in a byte)

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.12 Silicon Signature Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.12 Silicon Signature Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.12 Silicon Signature Command**.

5.11 Version Get Command

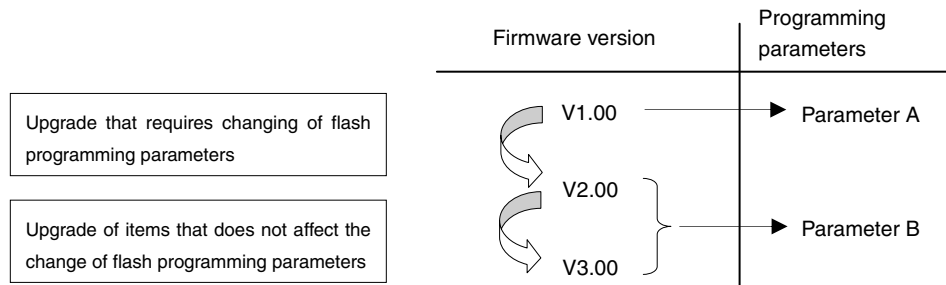
5.11.1 Description

This command is used to acquire information on the 78K0/Kx1+ device version and firmware version.

Use this command when the programming parameters must be changed in accordance with the 78K0/Kx1+ firmware version.

Caution The firmware version may be updated during firmware update that does not affect the change of flash programming parameters (at this time, update of the firmware version is not reported).

Example Firmware version and reprogramming parameters



5.11.2 Command frame and status frame

Figure 5-28 shows the format of a command frame for the Version Get command, and Figure 5-29 shows the status frame for the command.

Figure 5-28. Version Get Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	SUM	ETX
01H	01H	C5H (Version Get)	Checksum	03H

Figure 5-29. Status Frame for Version Get Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Command reception result

5.11.3 Version data frame

Figure 5-30 shows the data frame of version data.

Figure 5-30. Version Data Frame (from 78K0/Kx1+ to Programmer)

STX	LEN	Data						SUM	ETX
02H	06H	DV1	DV2	DV3	FV1	FV2	FV3	Checksum	03H

Remark DV1: Integer of device version
 DV2: First decimal place of device version
 DV3: Second decimal place of device version
 FV1: Integer of firmware version
 FV2: First decimal place of firmware version
 FV3: Second decimal place of firmware version

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.13 Version Get Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.13 Version Get Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.13 Version Get Command**.

5.12 Checksum Command

5.12.1 Description

This command is used to acquire the checksum data in the specified area.

For the checksum calculation start/end address, specify a fixed address in block units (2 KB) starting from the top of the flash memory.

Checksum data is obtained by sequentially subtracting data in the specified address range from the initial value (00H) in 1-byte units.

5.12.2 Command frame and status frame

Figure 5-31 shows the format of a command frame for the Checksum command, and Figure 5-32 shows the status frame for the command.

Figure 5-31. Checksum Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information						SUM	ETX
01H	07H	B0H (Checksum)	SAH	SAM	SAL	EAH	EAM	EAL	Checksum	03H

Remark SAH, SAM, SAL: Checksum calculation start addresses
EAH, EAM, EAL: Checksum calculation end addresses

Figure 5-32. Status Frame for Checksum Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1	Checksum	03H

Remark ST1: Command reception result

5.12.3 Checksum data frame

Figure 5-33 shows the format of a frame that includes checksum data.

Figure 5-33. Checksum Data Frame (from 78K0/Kx1+ to Programmer)

STX	LEN	Data		SUM	ETX
02H	02H	CK1	CK2	Checksum	03H

Remark CK1: Higher 8 bits of checksum data
CK2: Lower 8 bits of checksum data

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

- For the UART communication mode, read **6.14 Checksum Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.14 Checksum Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.14 Checksum Command**.

5.13 Security Set Command

5.13.1 Description

This command is used to perform security settings (enable or disable of write, block erase, and chip erase). By performing these settings with this command, rewriting of the flash memory by an unauthorized party can be restricted.

Caution Once the security setting is performed, additional setting to the security flags or changing of the setting from disable to enable will no longer be possible. If such settings are attempted, a Write error (1CH) will occur. To re-set the security flag, all the security flags must be initialized by executing the Chip Erase command (the Block Erase command cannot be used to initialize the security flags). If chip erase has been disabled, however, chip erase itself will be impossible and so the settings cannot be erased from the programmer. Re-confirmation of security setting execution is therefore recommended before disabling chip erase, due to this programmer specification.

5.13.2 Command frame and status frame

Figure 5-34 shows the format of a command frame for the Security Set command, and Figure 5-35 shows the status frame for the command.

The Security Set command frame includes the block number field and page number field but these fields do not have any particular usage, so set these fields to 00H.

Figure 5-34. Security Set Command Frame (from Programmer to 78K0/Kx1+)

SOH	LEN	COM	Command Information		SUM	ETX
01H	03H	A0H (Security Set)	BLK	PAG	Checksum	03H

Remark BLK, PAG: Fixed to 00H

Figure 5-35. Status Frame for Security Set Command (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (a)	Checksum	03H

Remark ST1 (a): Command reception result

5.13.3 Data frame and status frame

Figure 5-36 shows the format of a security data frame, and Figure 5-37 shows the status frame for the data.

Figure 5-36 Security Data Frame (from Programmer to 78K0/Kx1+)

STX	LEN	Data	SUM	ETX
02H	01H	FLG	Checksum	03H

Remark FLG: Security flag

Figure 5-37. Status Frame for Security Data Writing (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (b)	Checksum	03H

Remark ST1 (b): Security data write result

5.13.4 Internal verify check and status frame

Figure 5-38 shows the status frame for internal verify check.

Figure 5-38. Status Frame for Internal Verify Check (from 78K0/Kx1+ to Programmer)

STX	LEN	Data	SUM	ETX
02H	01H	ST1 (c)	Checksum	03H

Remark ST1 (c): Internal verify result

The following table shows the contents in the security flag field.

Table 5-2. Contents of Security Flag Field

Item	Contents
Bit 7	Fixed to "1"
Bit 6	
Bit 5	
Bit 4	
Bit 3	
Bit 2	Programming disable flag (1: Enables programming, 0: Disable programming)
Bit 1	Block erase disable flag (1: Enables block erase, 0: Disable block erase)
Bit 0	Chip erase disable flag (1: Enables chip erase, 0: Disable chip erase)

The following table shows the relationship between the security flag field settings and the enable/disable status of each operation.

Table 5-3. Security Flag Field and Enable/Disable Status of Each Operation

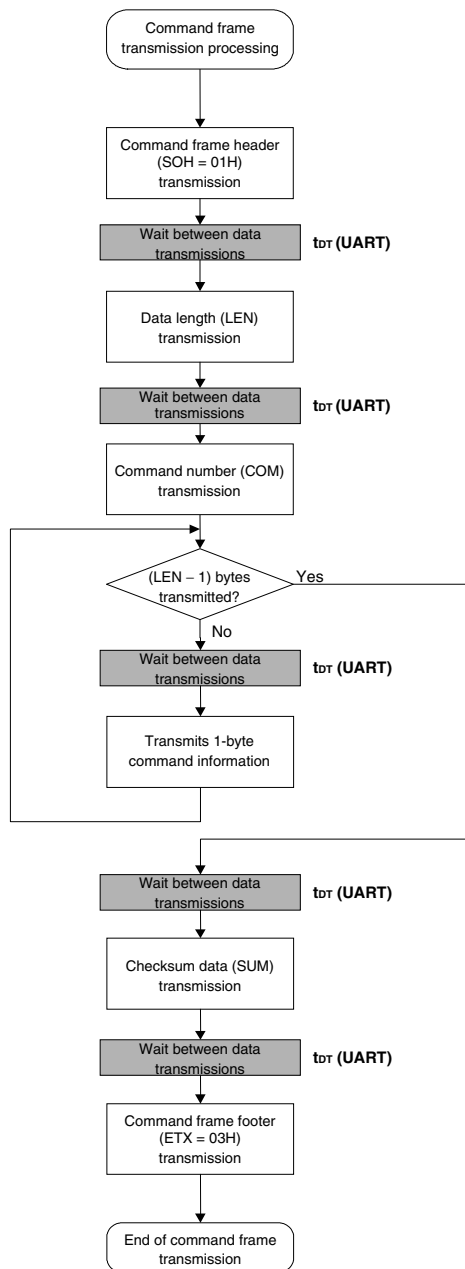
Operating Mode	Flash Memory Programming Mode			Self-Programming Mode	
Security Setting Item	Command Operation After Security Setting √: Execution possible, ×: Execution impossible			<ul style="list-style-type: none"> All commands can be executed regardless of the security setting values Only retention of security setting values is possible 	
	Programming	Chip Erase	Block Erase		
	Disable programming	×	√		×
	Disable chip erase	√	×		×
Disable block erase	√	√	×		

Read the following chapters for details on flowcharts of processing sequences between the programmer and the 78K0/Kx1+, flowcharts of command processing, and sample programs for each communication mode.

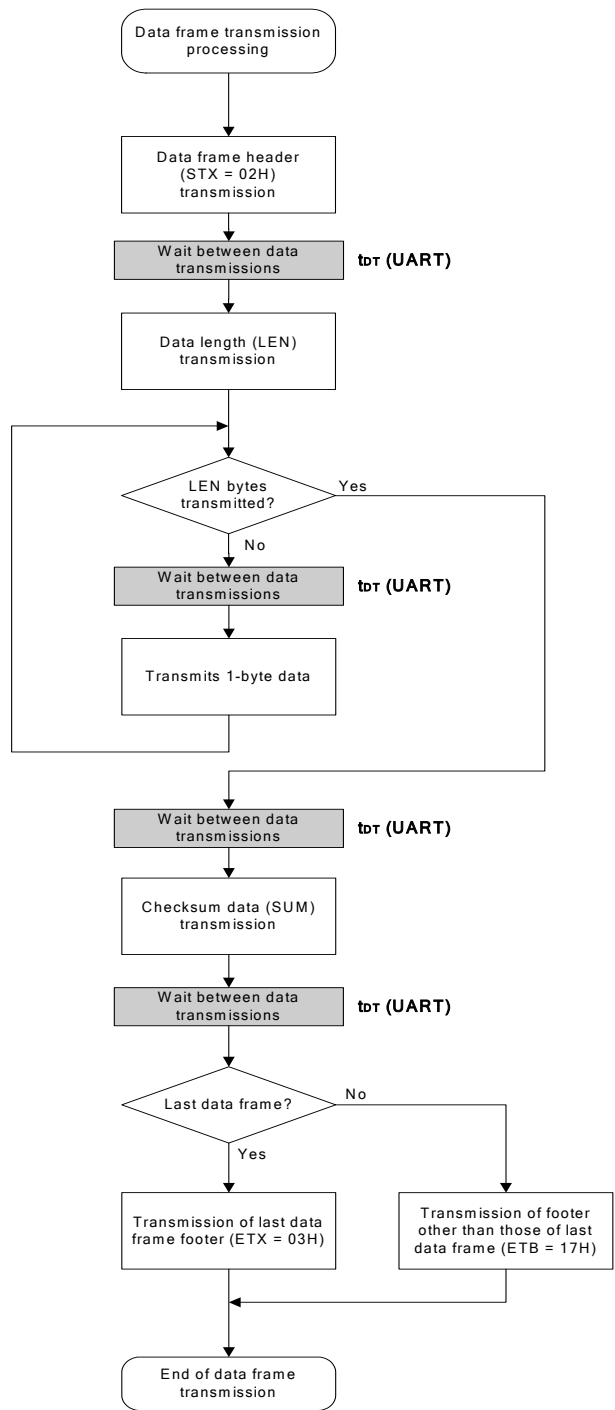
- For the UART communication mode, read **6.15 Security Set Command**.
- For the 3-wire serial I/O communication mode with handshake supported (CSI + HS), read **7.15 Security Set Command**.
- For the 3-wire serial I/O communication mode (CSI), read **8.15 Security Set Command**.

CHAPTER 6 UART COMMUNICATION MODE

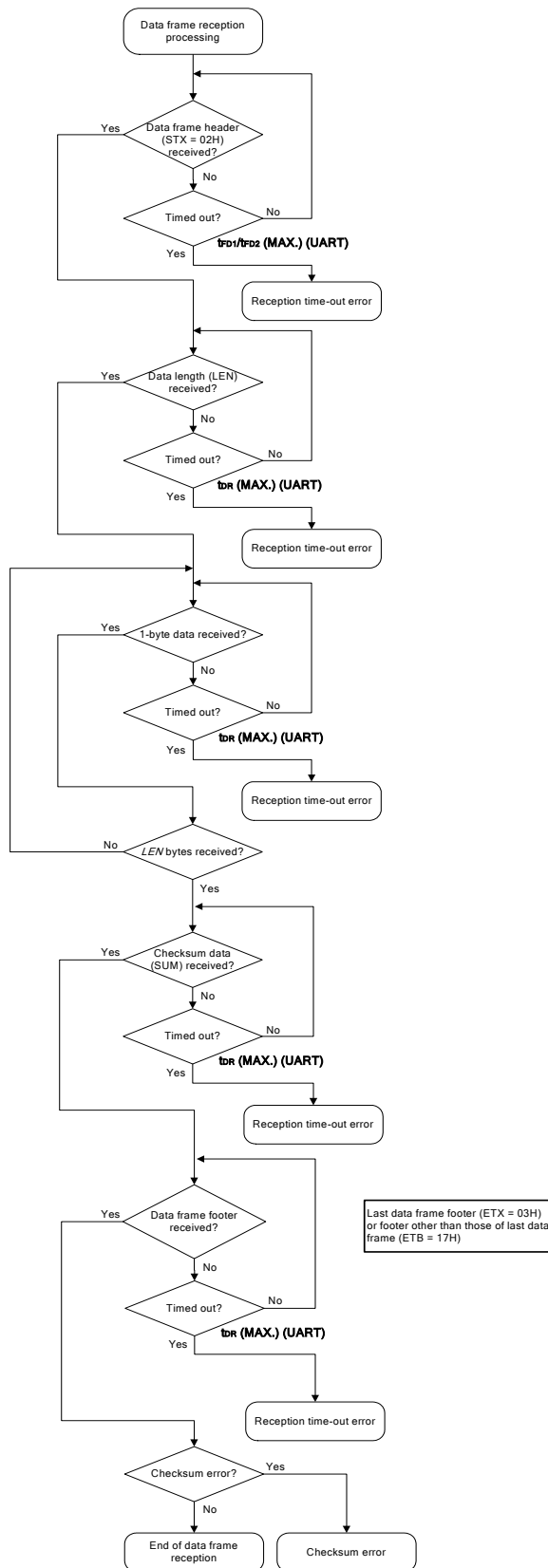
6.1 Command Frame Transmission Processing Flowchart



6.2 Data Frame Transmission Processing Flowchart



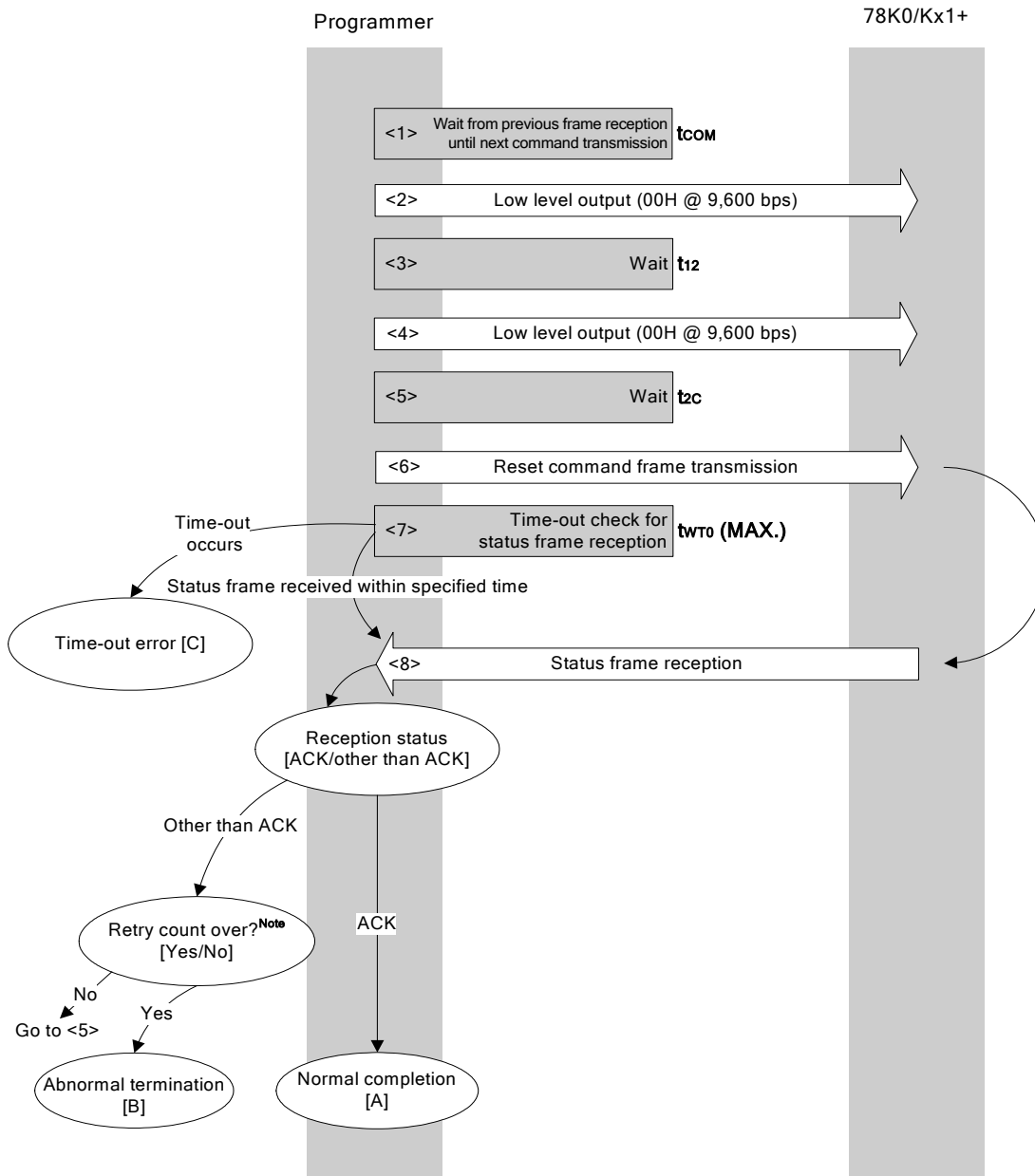
6.3 Data Frame Reception Processing Flowchart



6.4 Reset Command

6.4.1 Processing sequence chart

Reset command processing sequence



Note Do not exceed the retry count for the reset command transmission (up to 16 times).

6.4.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command processing starts (wait time t_{COM}).
- <2> The low level is output (data 00H is transmitted at 9,600 bps).
- <3> Wait state (wait time t_{12}).
- <4> The low level is output (data 00H is transmitted at 9,600 bps).
- <5> Wait state (wait time t_{2C}).
- <6> The Reset command is transmitted by command frame transmission processing.
- <7> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WTO} (MAX.)$).
- <8> The status code is checked.

When ST1 = ACK: Normal completion [A]

When ST1 \neq ACK: The retry count (t_{RS}) is checked.

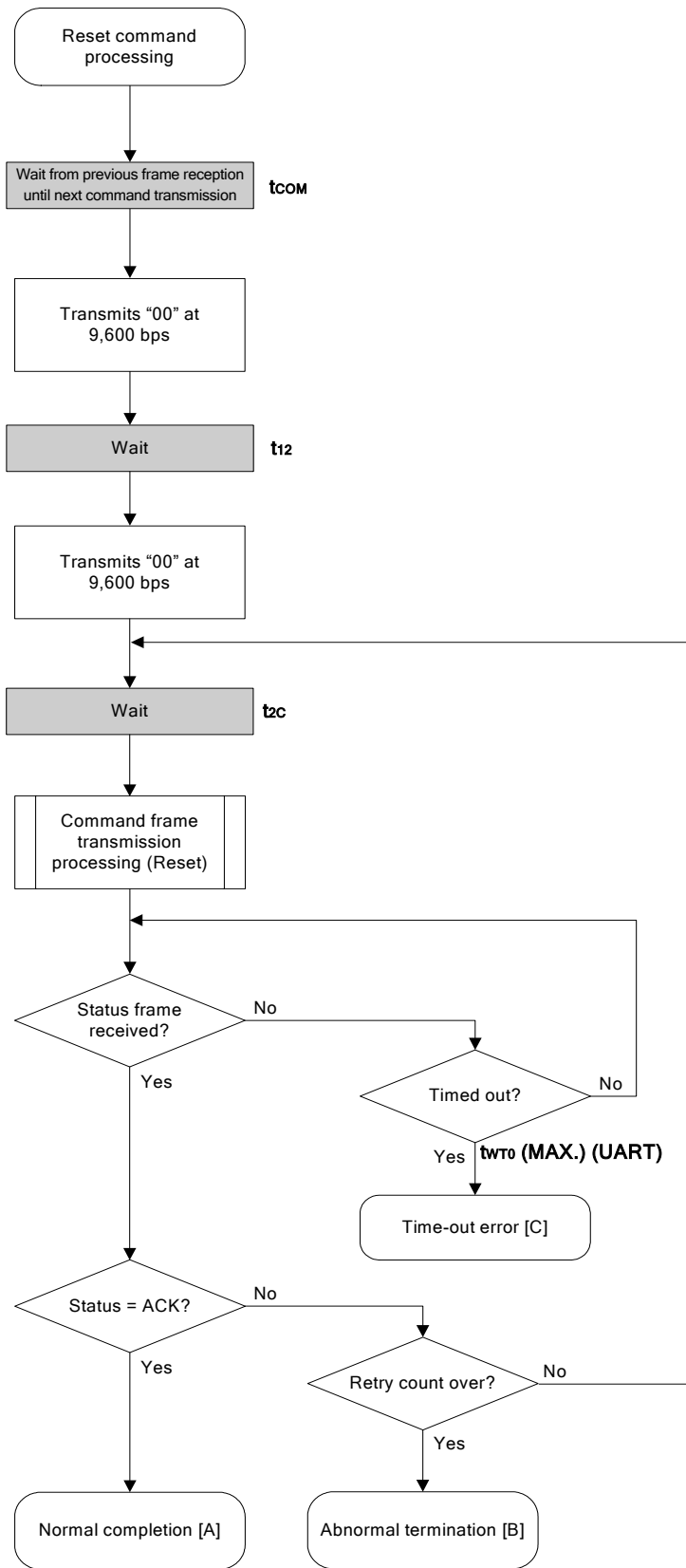
The sequence is re-executed from <5> if the retry count is not over.

If the retry count is over, the processing ends abnormally [B].

6.4.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and synchronization between the programmer and the 78K0/Kx1+ has been established.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.

6.4.4 Flowchart



6.4.5 Sample program

The following shows a sample program for Reset command processing.

```

/*****
/*
/* Reset command
/*
/*****
/* [r] u16      ... error code
/*****
u16 fl_ua_reset(void)
{
    u16    rc;
    u32    retry;

    set_uart0_br(BR_9600);    // change to 9600bps

    fl_wait(tCOM);           // wait
    putc_ua(0x00);           // send 0x00 @ 9600bps

    fl_wait(t12);            // wait
    putc_ua(0x00);           // send 0x00 @ 9600bps

    for (retry = 0; retry < tRS; retry++){

        fl_wait(t2C); // wait

        put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm);    // send RESET command

        rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX);
        if (rc == FLC_DFTO_ERR)    // t.o. ?
            break;                // yes // case [C]

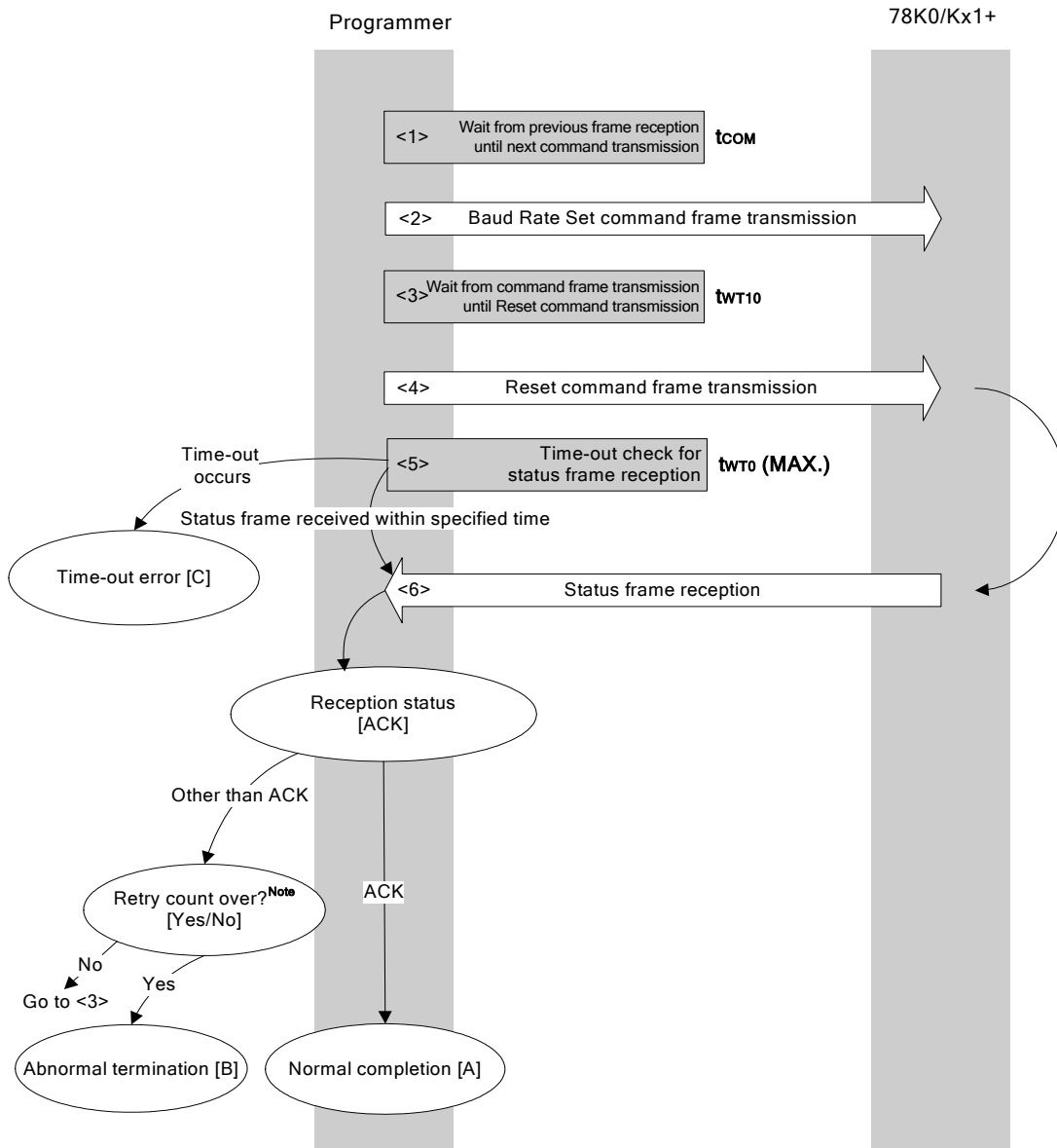
        if (rc == FLC_ACK){        // ACK ?
            break;                // yes // case [A]
        }
        else{
            NOP();
        }
        //continue;                // case [B] (if exit from loop)
    }
    // switch(rc) {
    //
    //     case  FLC_NO_ERR:  return rc;    break; // case [A]
    //     case  FLC_DFTO_ERR: return rc;    break; // case [C]
    //     default:          return rc;    break; // case [B]
    // }
    return rc;
}

```

6.5 Baud Rate Set Command

6.5.1 Processing sequence chart

Baud Rate Set command processing sequence



Note Do not exceed the retry count for the reset command transmission (up to 16 times).

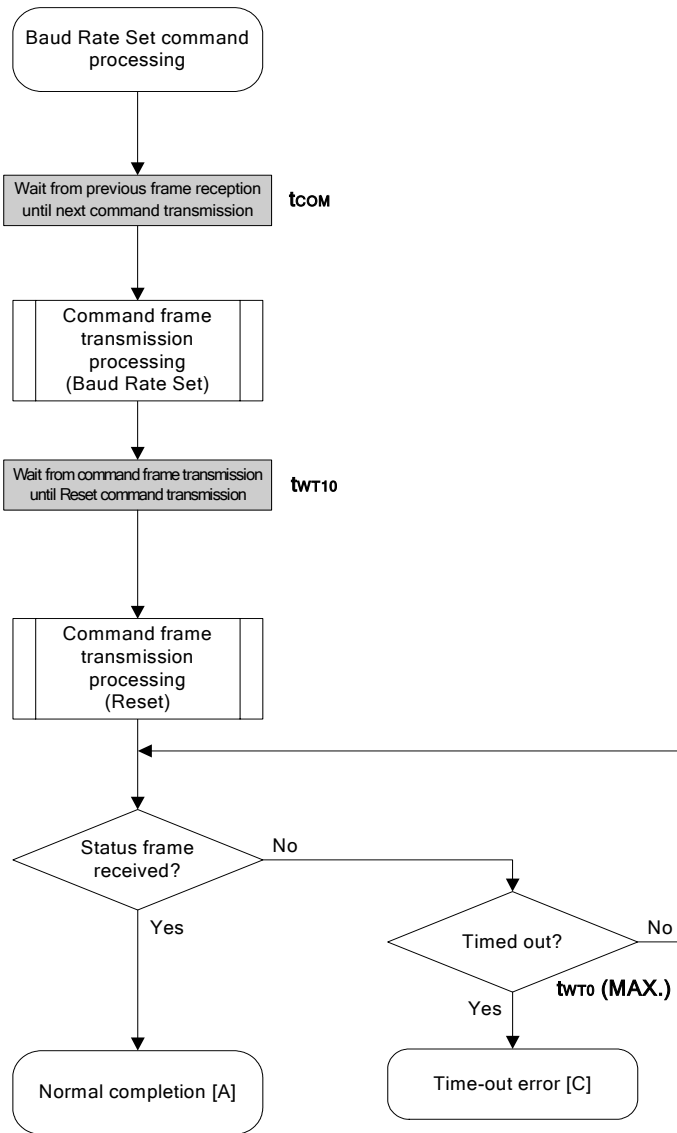
6.5.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Baud Rate Set command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until Reset command transmission (wait time t_{WT10}).
- <4> The Reset command is transmitted by command frame transmission processing.
- <5> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WTO} (MAX.)$).
- <6> Since the status code should be ACK, the processing ends normally [A].

6.5.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the synchronization of the UART communication speed has been established between the programmer and the 78K0/Kx1+.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Data frame reception was timed out. With the 78K0/Kx1+, this command also results in errors in the following cases. <ul style="list-style-type: none"> • Command information (parameter) is invalid • The command frame includes the checksum error • The data length of the command frame (LEN) is invalid • The footer of the command frame (ETX) is missing

6.5.4 Flowchart



6.5.5 Sample program

The following shows a sample program for Baud Rate Set command processing.

```

/*****
/*
/* Set baudrate command
/*
/*****
/* [i] u8 brid ... baudrate ID
/* [r] u16 ... error code
/*****
u16 fl_ua_setbaud(u8 brid)
{
    u16 rc;
    u8 br;
    u32 retry;

    switch(brid){
        default:
            case BR_9600: br = 0x03; break;
            case BR_19200: br = 0x04; break;
            case BR_31250: br = 0x05; break;
            case BR_38400: br = 0x06; break;
            case BR_76800: br = 0x07; break;
            case BR_153600: br = 0x08; break;
    }
    fl_cmd_prm[0] = br; // "D01"

    fl_wait(tCOM); // wait before sending command
    put_cmd_ua(FL_COM_SET_BAUDRATE, 2, fl_cmd_prm); // send "Baudrate Set" command

    set_flbaud(brid); // change baud-rate
    set_uart0_br(brid); // change baud-rate (h.w.)

    retry = tRS;
    while(1){
        fl_wait(tWT10);

        put_cmd_ua(FL_COM_RESET, 1, fl_cmd_prm); // send RESET command

        rc = get_sfrm_ua(fl_ua_sfrm, tWT0_MAX); // get status frame
        if (rc){
            if (retry--){
                continue;
            }
            else{
                return rc;
            }
        }
    }
}

```

```
        break;          // got ACK !!

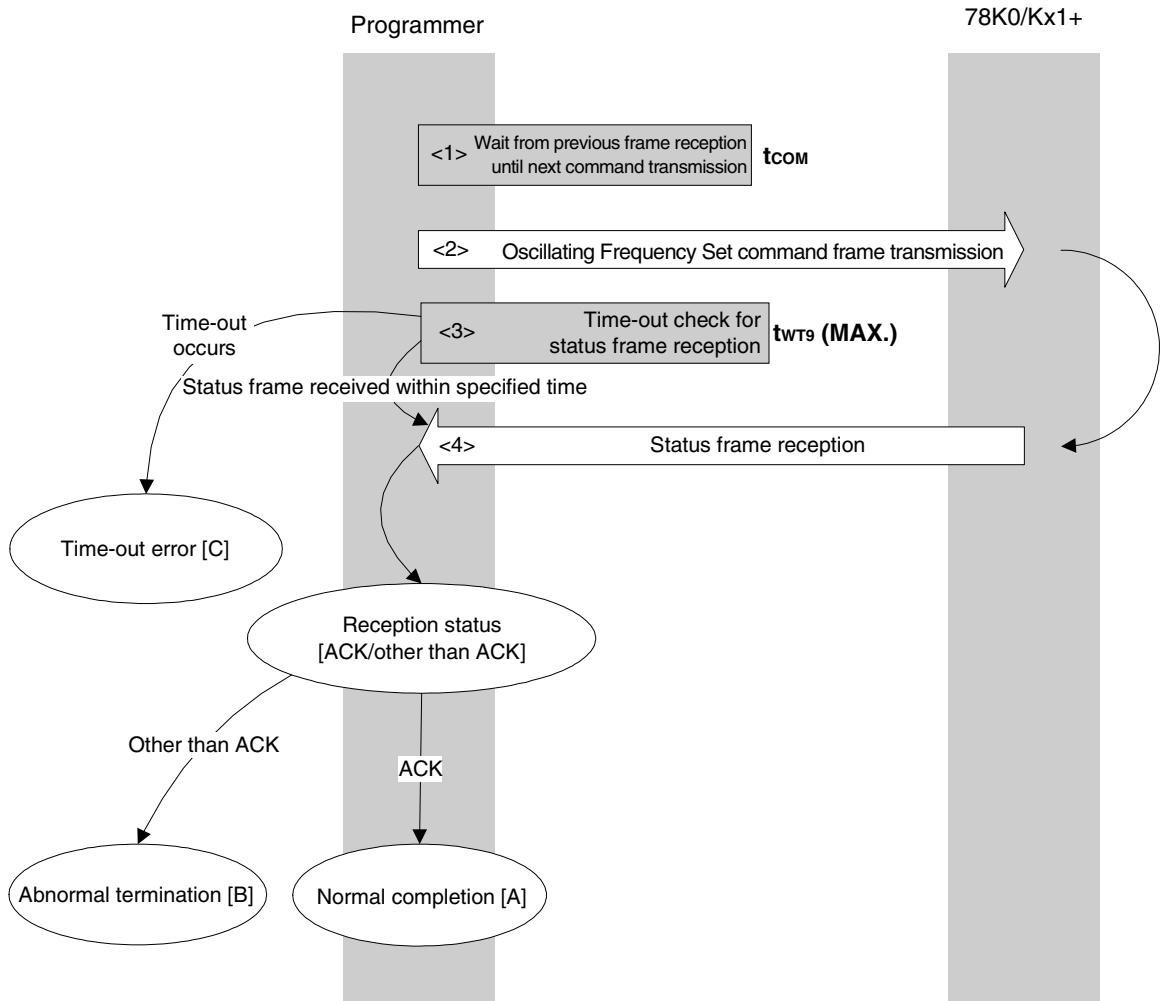
    }
// switch(rc) {
//     case FLC_NO_ERR:  return rc;   break; // case [A]
//     case FLC_DFTO_ERR: return rc;  break; // case [C]
//     default:         return rc;   break; // case [B]
// }

    return rc;
}
```


6.6 Oscillating Frequency Set Command

6.6.1 Processing sequence chart

Oscillating Frequency Set command processing sequence



6.6.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Oscillating Frequency Set command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT9} (MAX.)$).
- <4> The status code is checked.

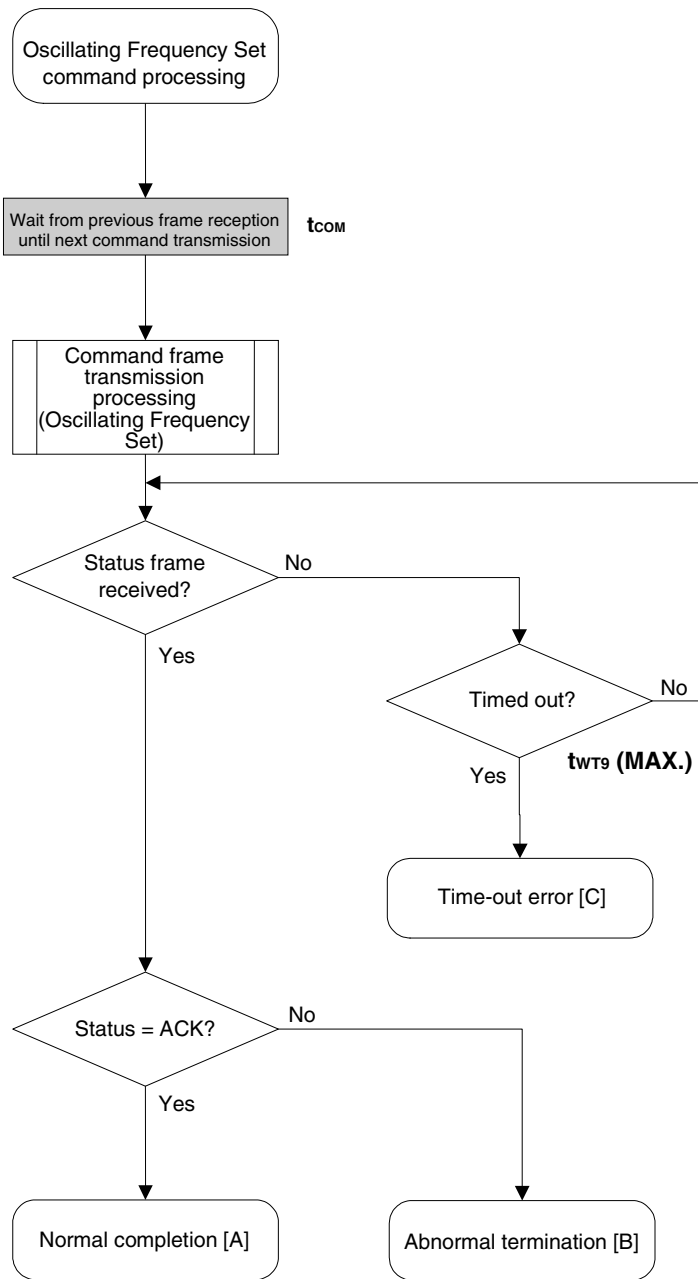
When ST1 = ACK: Normal completion [A]

When ST1 \neq ACK: Abnormal termination [B]

6.6.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the operating frequency was correctly set to the 78K0/Kx1+.
Abnormal termination [B]	Parameter error	05H	The oscillation frequency value is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.

6.6.4 Flowchart



6.6.5 Sample program

The following shows a sample program for Oscillating Frequency Set command processing.

```

/*****
/*
/* Set Flash device clock value command
/*
/*
/*****
/* [i] u8 clk[4] ... frequency data(D1-D4)
/* [r] u16 ... error code
/*****
u16 fl_ua_setclk(u8 clk[])
{
    u16 rc;

    fl_cmd_prm[0] = clk[0]; // "D01"
    fl_cmd_prm[1] = clk[1]; // "D02"
    fl_cmd_prm[2] = clk[2]; // "D03"
    fl_cmd_prm[3] = clk[3]; // "D04"

    fl_wait(tCOM); // wait before sending command
    put_cmd_ua(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);

    rc = get_sfrm_ua(fl_ua_sfrm, tWT9_MAX); // get status frame
    // switch(rc) {
    //
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }

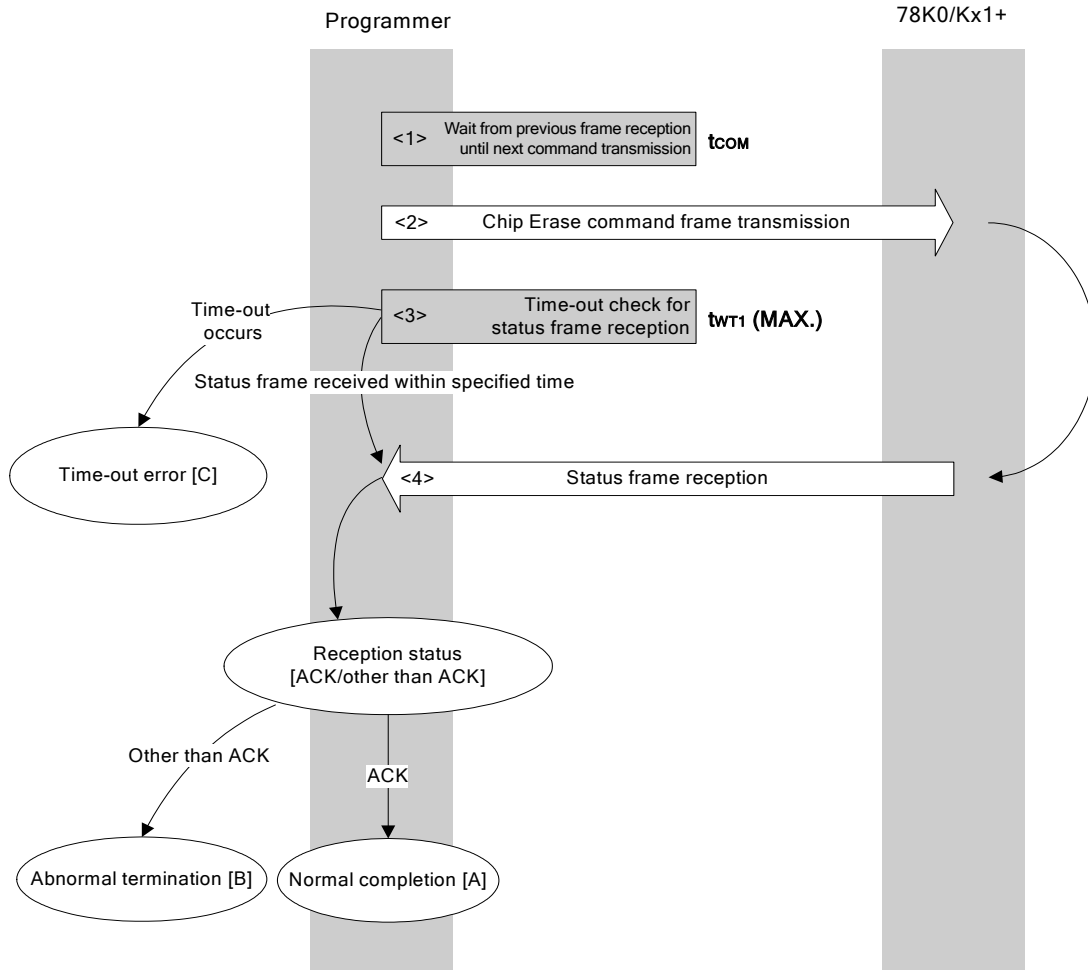
    return rc;
}

```

6.7 Chip Erase Command

6.7.1 Processing sequence chart

Chip Erase command processing sequence



6.7.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Chip Erase command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time t_{WT1} (MAX.)).
- <4> The status code is checked.

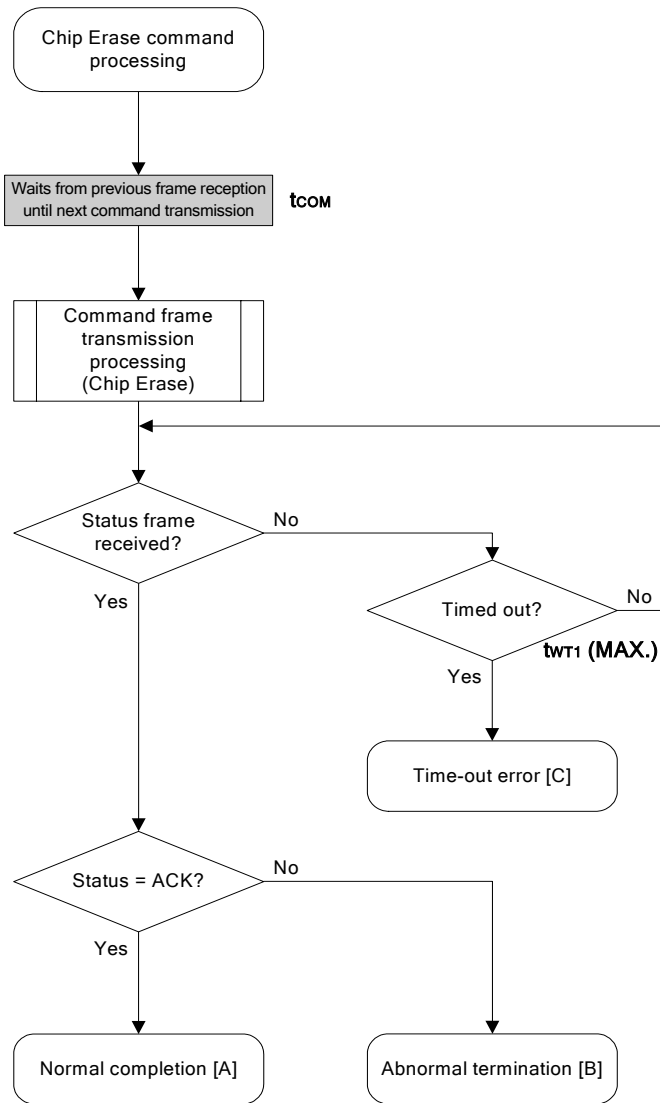
When ST1 = ACK: Normal completion [A]

When ST1 \neq ACK: Abnormal termination [B]

6.7.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and chip erase was performed normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Chip erase is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Erase error	1AH	An erase error has occurred.
Time-out error [C]		–	The status frame was not received within the specified time.

6.7.4 Flowchart



6.7.5 Sample program

The following shows a sample program for Chip Erase command processing.

```
/*
 *
 * Erase all(chip) command
 *
 */
/* [r] u16 ... error code */
u16 fl_ua_erase_all(void)
{
    u16 rc;

    fl_wait(tCOM); // wait before sending command

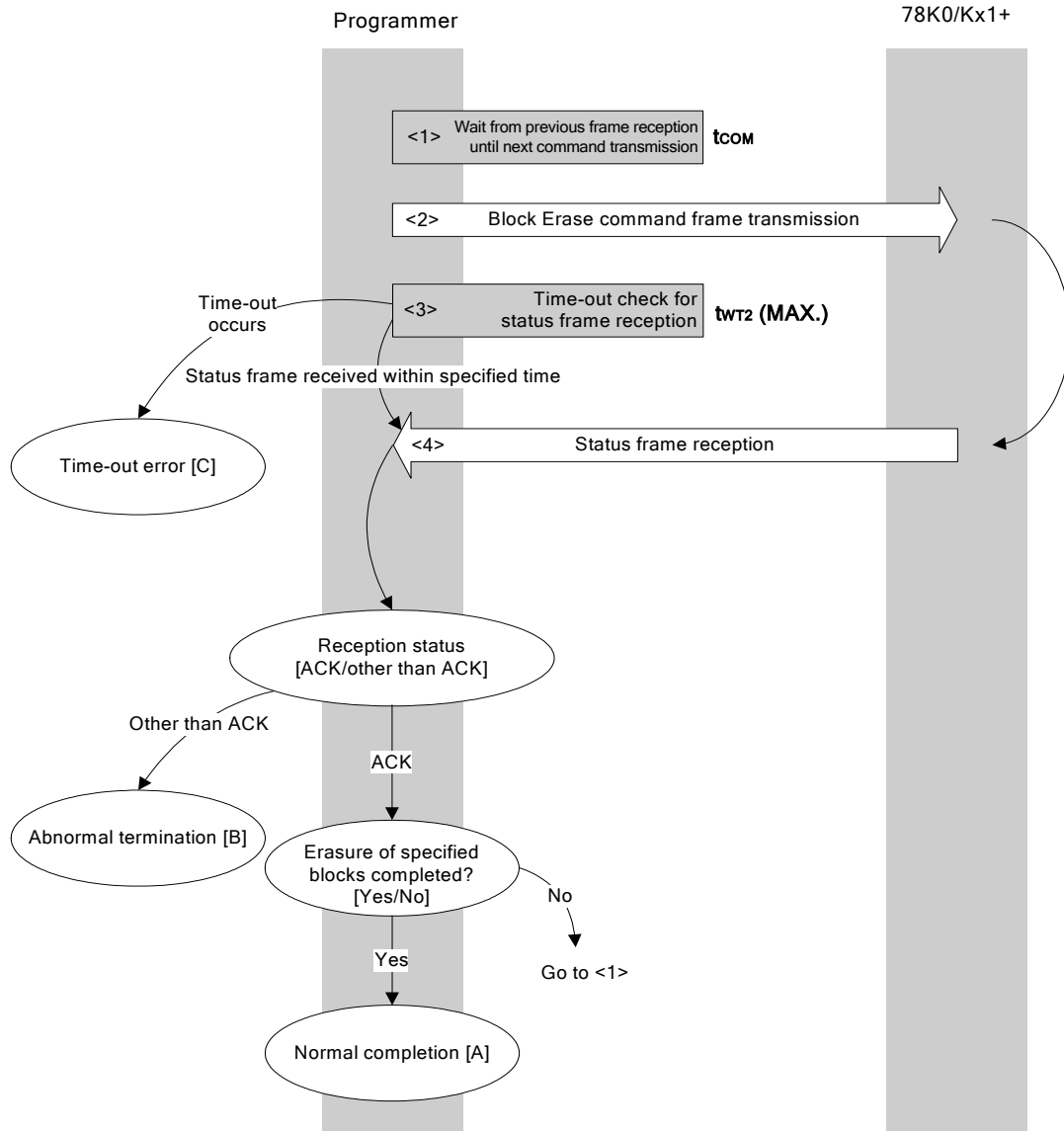
    put_cmd_ua(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // send ERASE CHIP command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT1_MAX); // get status frame
    // switch(rc) {
    //
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}
```


6.8 Block Erase Command

6.8.1 Processing sequence chart

Block Erase command processing sequence



6.8.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Block Erase command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WTT2} (MAX.)$).
- <4> The status code is checked.

When ST1 = ACK: When the block erase for all of the specified blocks is not yet completed, processing changes the block number and re-executes the sequence from <1>.

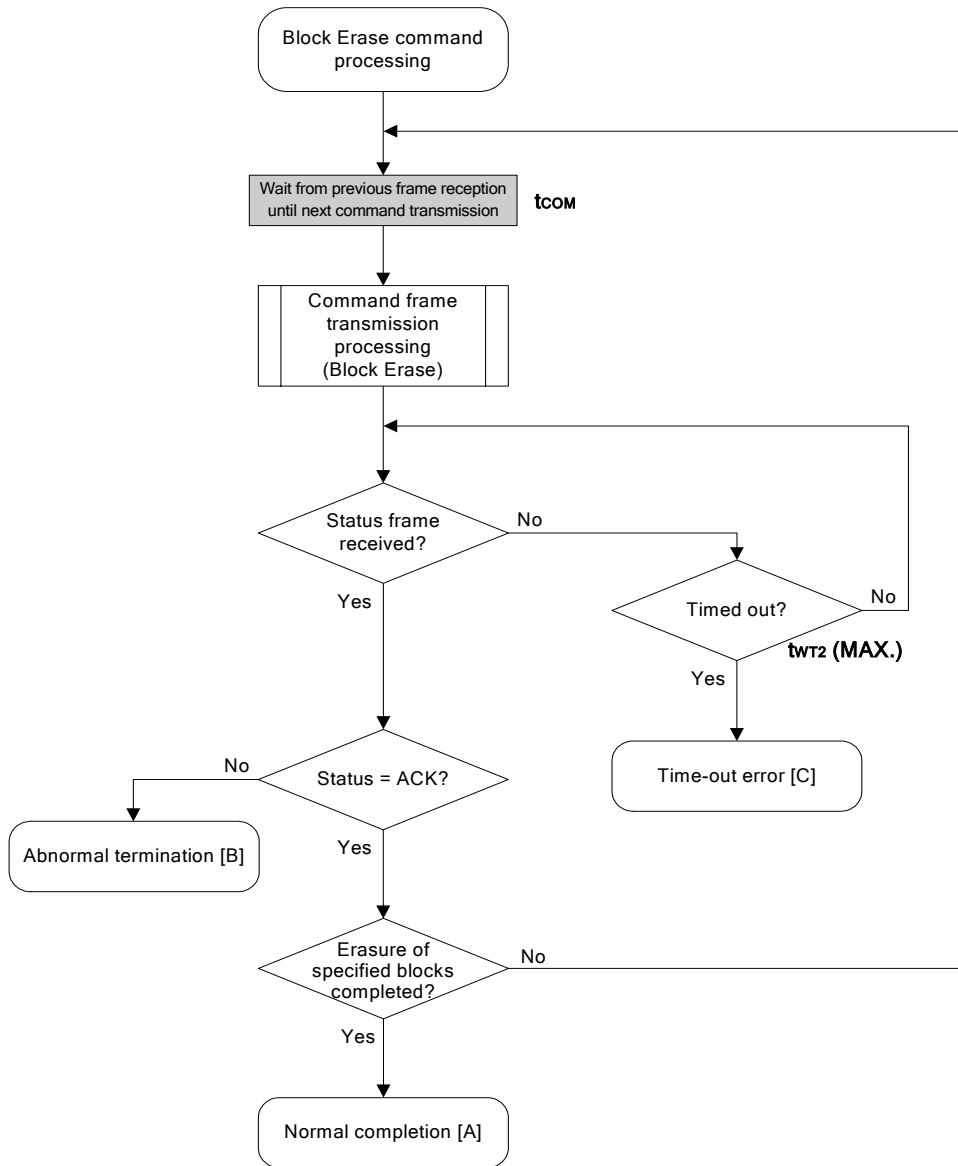
When the block erase for all of the specified blocks is completed, the processing ends normally [A].

When ST1 \neq ACK: Abnormal termination [B]

6.8.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and block erase was performed normally.
Abnormal termination [B]	Parameter error	05H	The block number is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Write, block erase, or chip erase is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Erase error	1AH	An erase error has occurred.
Time-out error [C]		–	The status frame was not received within the specified time.

6.8.4 Flowchart



6.8.5 Sample program

The following shows a sample program for Block Erase command processing for one block.

```

/*****
/*
/* Erase block command
/*
/*****
/* [i] u8 block    ... block number
/* [r] u16         ... error code
/*****
u16      fl_ua_erase_blk(u8 block)
{
    u16    rc;

    fl_cmd_prm[0] = block;    // BLK

    fl_wait(tCOM);           // wait before sending command

    put_cmd_ua(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm); // send ERASE CHIP command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT2_MAX); // get status frame
//    switch(rc) {
//
//        case FLC_NO_ERR:    return rc;    break; // case [A]
//        case FLC_DFTO_ERR:  return rc;    break; // case [C]
//        default:            return rc;    break; // case [B]
//    }

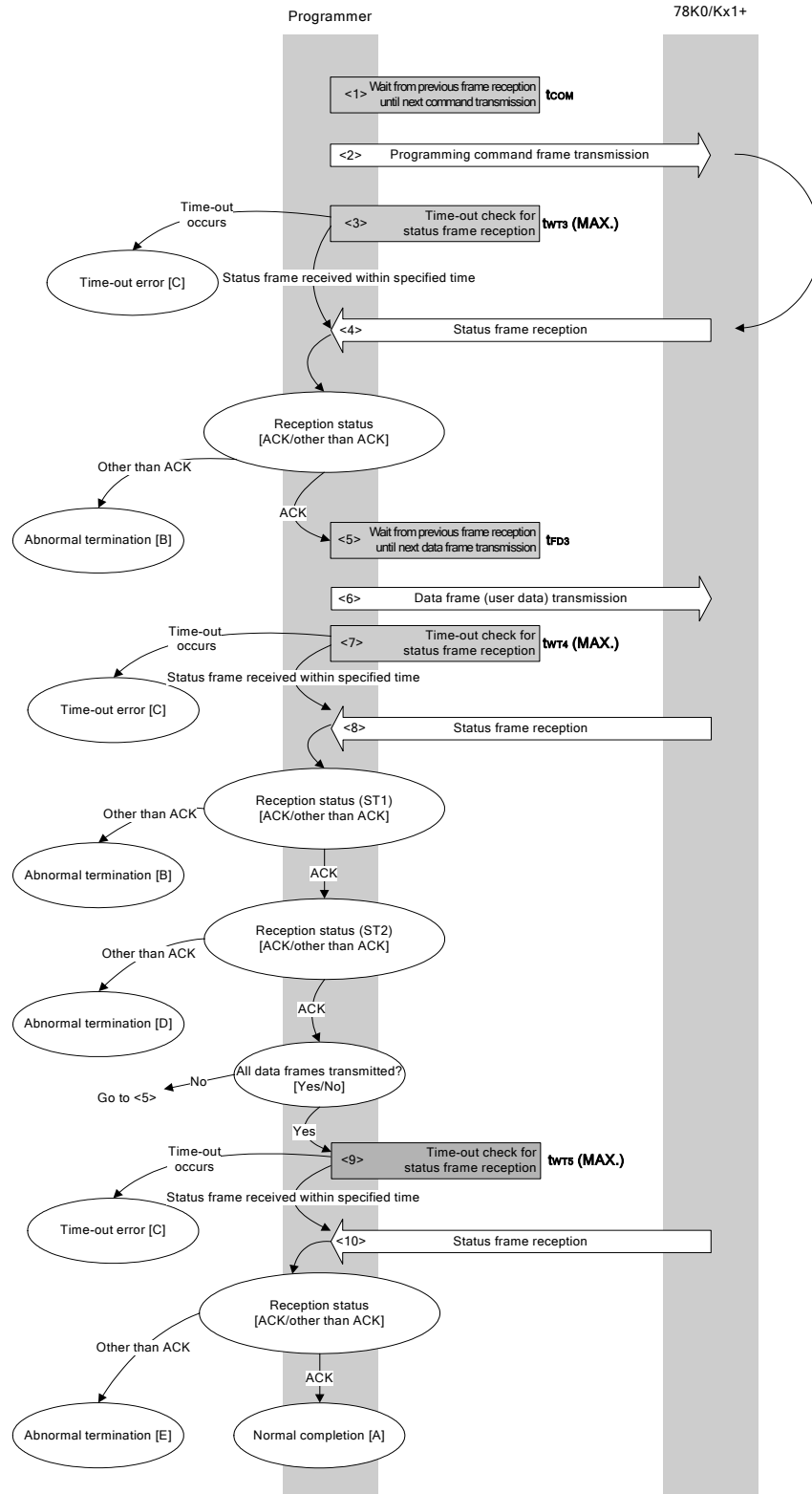
    return rc;
}

```

6.9 Programming Command

6.9.1 Processing sequence chart

Programming command processing sequence



6.9.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Programming command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT3} (MAX.)$).
- <4> The status code is checked.

When ST1 = ACK: Proceeds to <5>.

When ST1 \neq ACK: Abnormal termination [B]

- <5> Waits from the previous frame reception until the next data frame transmission (wait time t_{FD3}).
- <6> User data is transmitted by data frame transmission processing.
- <7> A time-out check is performed from user data transmission until data frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT4} (MAX.)$).
- <8> The status code (ST1/ST2) is checked (also refer to the processing sequence chart and flowchart).

When ST1 \neq ACK: Abnormal termination [B]

When ST1 = ACK: The following processing is performed according to the ST2 value.

- When ST2 = ACK: Proceeds to <9> when transmission of all data frames is completed.
If there still remain data frames to be transmitted, the processing re-executes the sequence from <5>.
- When ST2 \neq ACK: Abnormal termination [D]

- <9> A time-out check is performed until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT5} (MAX.)$).
- <10> The status code is checked.

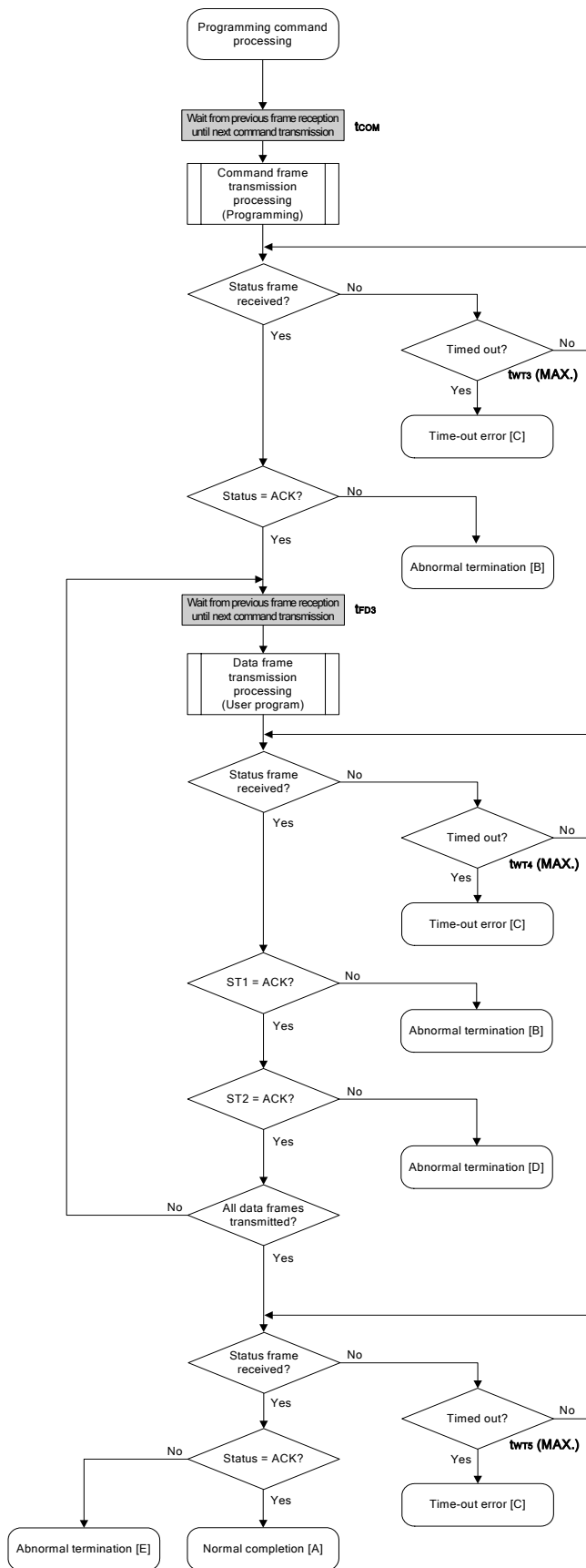
When ST1 = ACK: Normal completion [A]

When ST1 \neq ACK: Abnormal termination [E]

6.9.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the user data was written normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or is not a multiple of 8.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Write is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.
Abnormal termination [D]	FLMD error	18H (ST2)	A write error has occurred.
	Write error	1CH (ST2)	A write error has occurred.
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

6.9.4 Flowchart



6.9.5 Sample program

The following shows a sample program for Programming command processing.

```

/*****
/*
/* Write command
/*
/*****
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****

#define          fl_st2_ua      (fl_ua_sfrm[OFS_STA_PLD+1])

u16 fl_ua_write(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;
    u16    block_num;

    /*****
    /*      set params
    /*****
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    block_num = get_block_num(top, bottom); // get block num

    /*****
    /*      send command & check status
    /*****
    fl_wait(tCOM);           // wait before sending command

    put_cmd_ua(FL_COM_WRITE, 7, fl_cmd_prm); // send "Programming" command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT3_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR:           break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:                   return rc; break; // case [B]
    }

    /*****
    /*      send user data
    /*****
    send_head = top;

```

```

while(1){

// make send data frame
if ((bottom - send_head) > 256){ // rest size > 256 ?
    is_end = false;           // yes, not is_end frame
    send_size = 256;         // transmit size = 256 byte
}
else{
    is_end = true;
    send_size = bottom - send_head + 1; // transmit size = (bottom -
                                        // send_head)+1 byte
}

memcpy(fl_txdata_frm, rom_buf+send_head, send_size); // set data frame
                                                    // payload

send_head += send_size;

fl_wait(tFD3); // wait before sending data frame

put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data

rc = get_sfrm_ua(fl_ua_sfrm, tWT4_MAX); // get status frame
switch(rc) {
    case FLC_NO_ERR:           break; // continue
    case FLC_DFTO_ERR: return rc; break; // case [C]
    default:                   return rc; break; // case [B]
}
if (fl_st2_ua != FLST_ACK){ // ST2 = ACK ?
    rc = decode_status(fl_st2_ua); // No
    return rc; // case [D]
}
if (is_end)
    break;
}

/*****
/*    Check internally verify    */
*****/

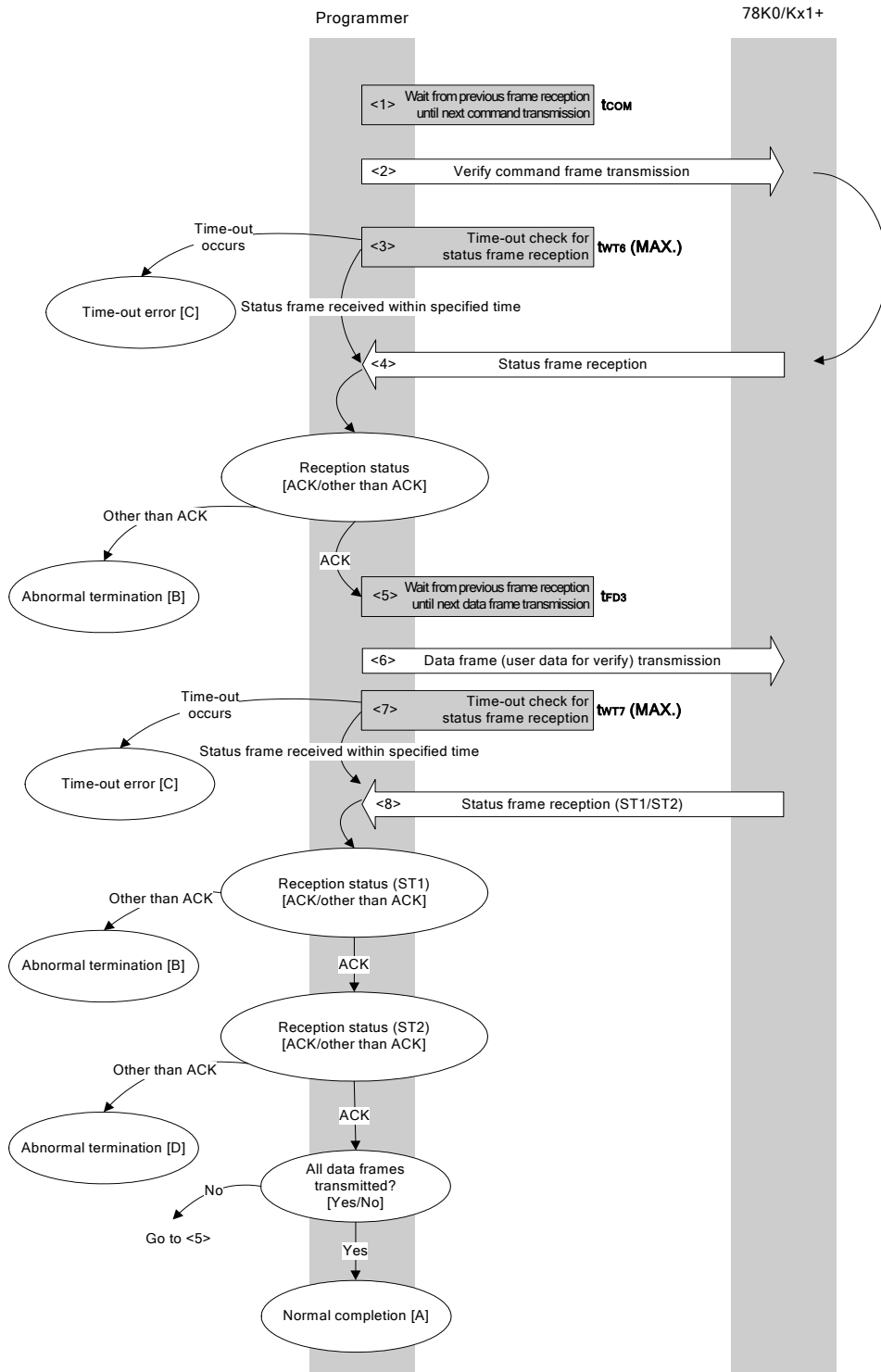
rc = get_sfrm_ua(fl_ua_sfrm, (tWT5_MAX * block_num)); //get status frame again
// switch(rc) {
//     case FLC_NO_ERR:  return rc;  break; // case [A]
//     case FLC_DFTO_ERR: return rc;  break; // case [C]
//     default:         return rc;  break; // case [E]
// }
return rc;
}

```

6.10 Verify Command

6.10.1 Processing sequence chart

Verify command processing sequence



6.10.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Verify command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT6} (MAX.)$).
- <4> The status code is checked.

When ST1 = ACK: Proceeds to <5>.

When ST1 \neq ACK: Abnormal termination [B]

- <5> Waits from the previous frame reception until the next data frame transmission (wait time t_{FD3}).
- <6> User data for verifying is transmitted by data frame transmission processing.
- <7> A time-out check is performed from user data transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT7} (MAX.)$).
- <8> The status code (ST1/ST2) is checked (also refer to the processing sequence chart and flowchart).

When ST1 \neq ACK: Abnormal termination [B]

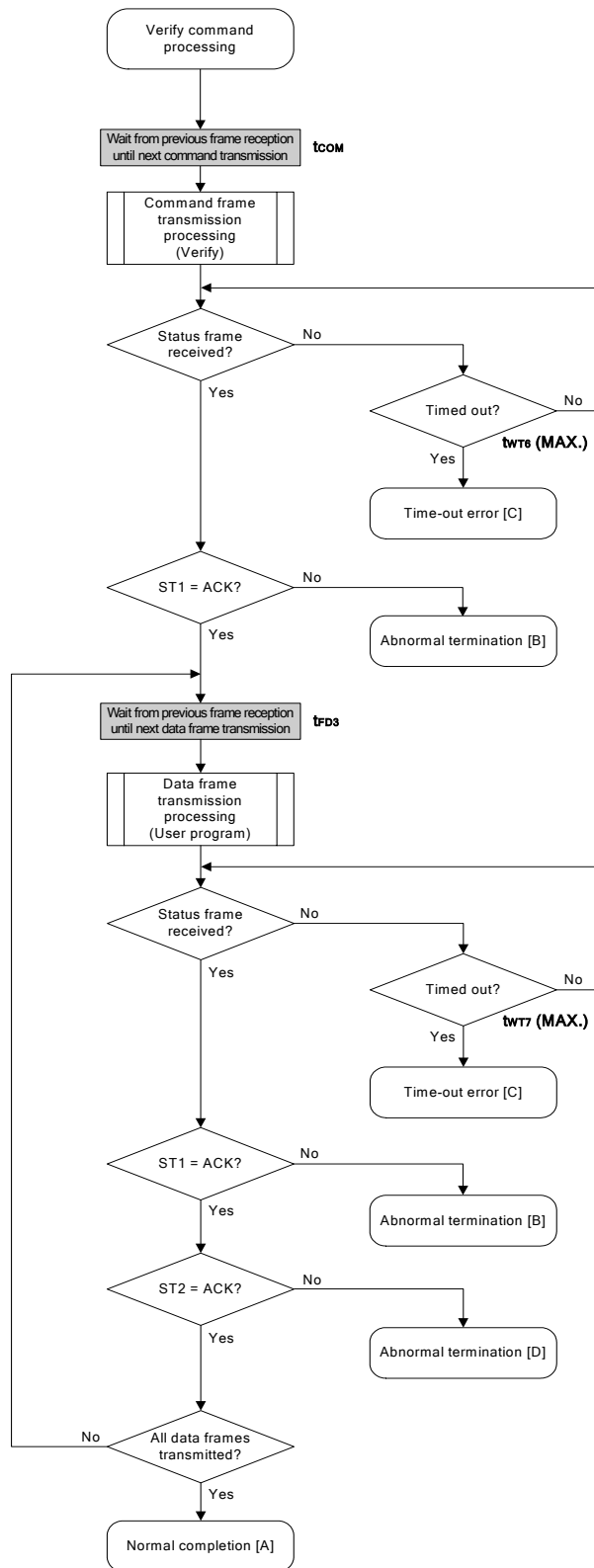
When ST1 = ACK: The following processing is performed according to the ST2 value.

- When ST2 = ACK: If transmission of all data frames is completed, the processing ends normally [A].
If there still remain data frames to be transmitted, the processing re-executes the sequence from <5>.
- When ST2 \neq ACK: Abnormal termination [D]

6.10.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the verify was completed normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame or data frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.
Abnormal termination [D]	Verify error	0FH (ST2)	The verify has failed, or another error has occurred.

6.10.4 Flowchart



6.10.5 Sample program

The following shows a sample program for Verify command processing.

```

/*****
/*
/* Verify command
/*
/*****
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****
u16 fl_ua_verify(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;

    /*****
    /*      set params
    /*****
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    /*****
    /*      send command & check status
    /*****

    fl_wait(tCOM);           // wait before sending command

    put_cmd_ua(FL_COM_VERIFY, 7, fl_cmd_prm); // send VERIFY command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT6_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR:           break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:                   return rc; break; // case [B]
    }

    /*****
    /*      send user data
    /*****
    send_head = top;

    while(1){

        // make send data frame
        if ((bottom - send_head) > 256){ // rest size > 256 ?

```

```

        is_end = false;    // yes, not is_end frame
        send_size = 256;  // transmit size = 256 byte
    }
    else{
        is_end = true;
        send_size = bottom - send_head + 1;    // transmit size =
                                                // (bottom - send_head)+1 byte
    }

    memcpy(fl_txdata_frm, rom_buf+send_head, send_size); // set data
                                                         // frame payload

    send_head += send_size;

    fl_wait(tFD3);
    put_dfrm_ua(send_size, fl_txdata_frm, is_end); // send user data

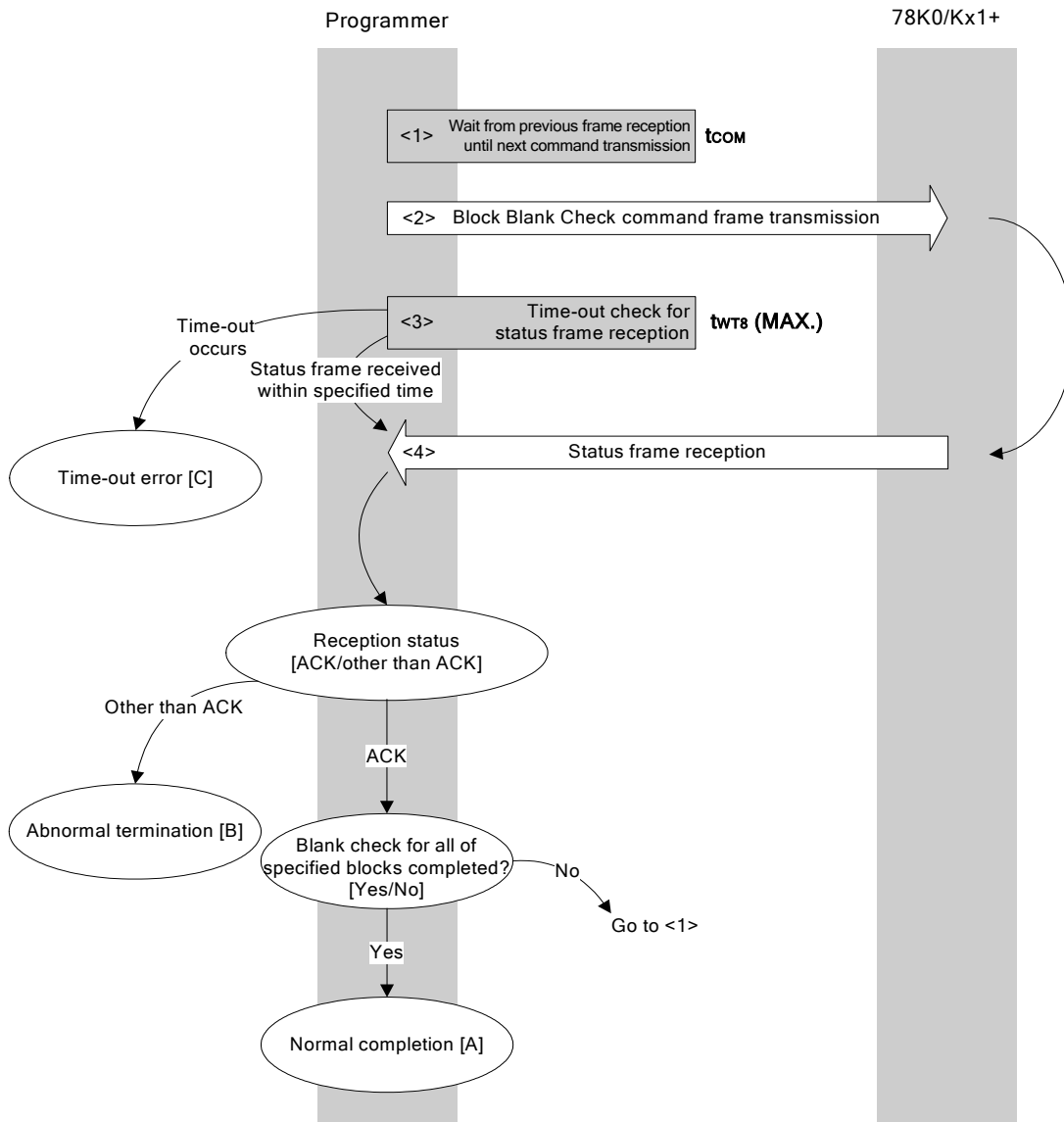
    rc = get_sfrm_ua(fl_ua_sfrm, tWT7_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:                        return rc; break; // case [B]
    }
    if (fl_st2_ua != FLST_ACK){          // ST2 = ACK ?
        rc = decode_status(fl_st2_ua);  // No
        return rc;                      // case [D]
    }
    if (is_end)                          // send all user data ?
        break;                          // yes
    //continue;
}
return FLC_NO_ERR; // case [A]
}

```

6.11 Block Blank Check Command

6.11.1 Processing sequence chart

Block Blank Check command processing sequence



6.11.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Block Blank Check command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT8} (MAX.)$).
- <4> The status code is checked.

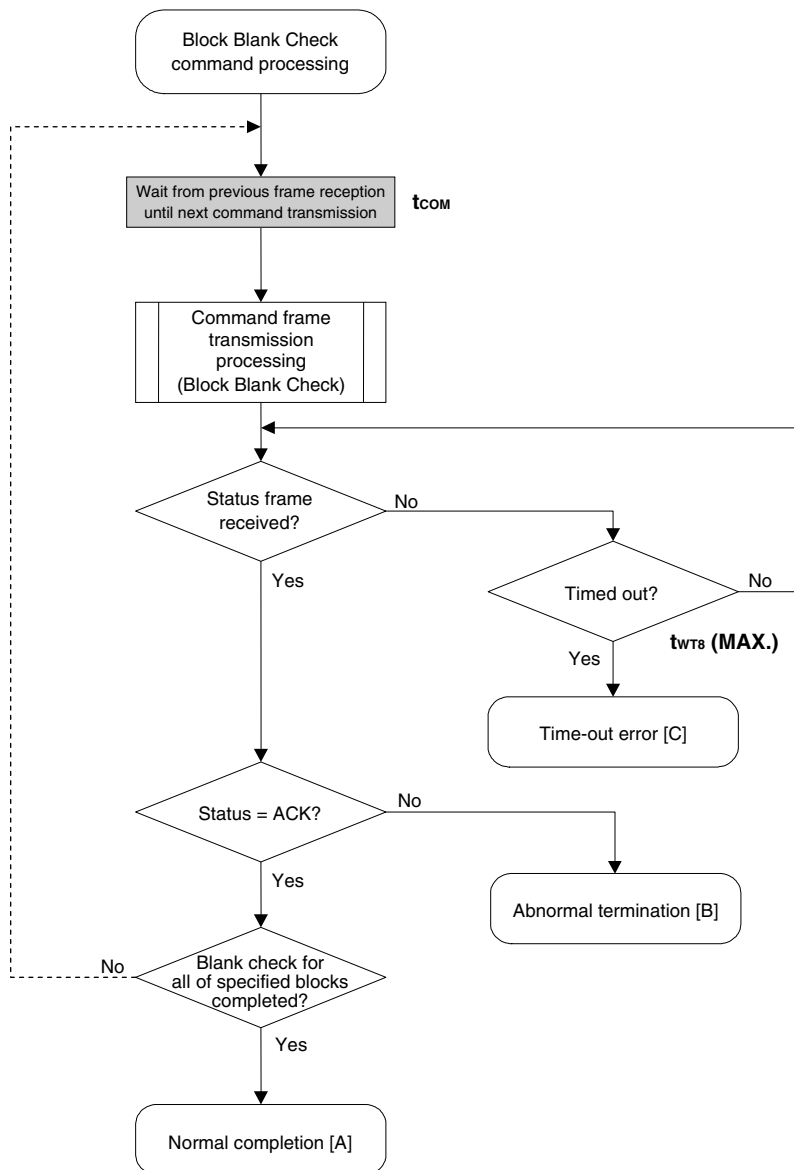
When ST1 = ACK: If the blank check for all of the specified blocks is not yet completed, processing changes the block number and re-executes the sequence from <1>.
If the blank check for all of the specified blocks is completed, the processing ends normally [A].

When ST1 \neq ACK: Abnormal termination [B]

6.11.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and all of the specified blocks are blank.
Abnormal termination [B]	Parameter error	05H	The block number is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	MRG11 error	1BH	The specified block in the flash memory is not blank.
Time-out error [C]		–	The status frame was not received within the specified time.

6.11.4 Flowchart



6.11.5 Sample program

The following shows a sample program for Block Blank Check command processing for one block.

```

/*****
/*
/* Block blank check command
/*
/*****
/* [i] u8 block ... block number
/* [r] u16 ... error code
/*****
u16 fl_ua_blk_blank_chk(u8 block)
{
    u16 rc;

    fl_cmd_prm[0] = block; // "BLK"

    fl_wait(tCOM); // wait before sending command

    put_cmd_ua(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm);

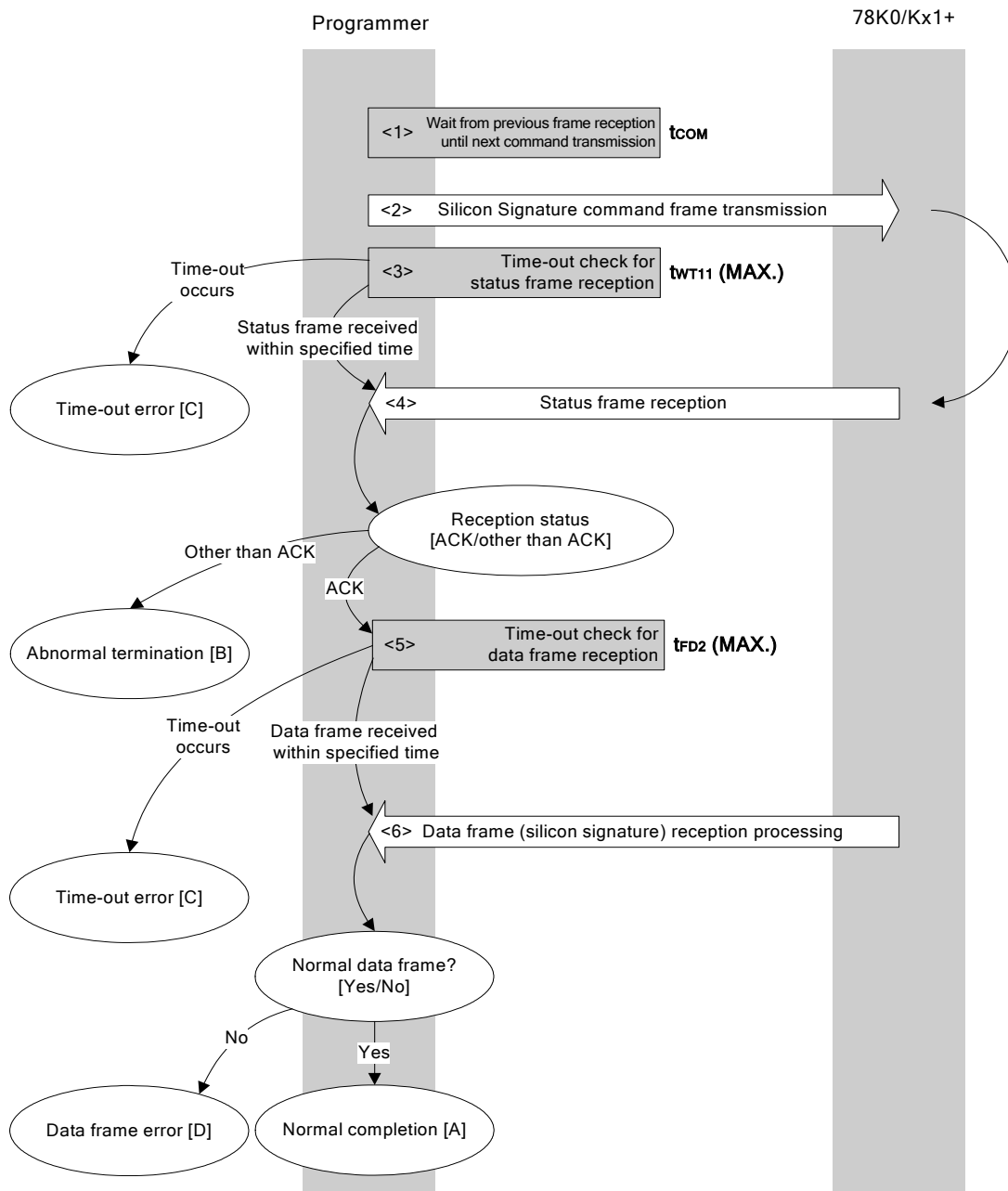
    rc = get_sfrm_ua(fl_ua_sfrm, tWT8_MAX); // get status frame
    // switch(rc) {
    //
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}

```

6.12 Silicon Signature Command

6.12.1 Processing sequence chart

Silicon Signature command processing sequence



6.12.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Silicon Signature command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT11} (MAX.)$).
- <4> The status code is checked.

When ST1 = ACK: Proceeds to <5>.

When ST1 \neq ACK: Abnormal termination [B]

- <5> A time-out check is performed until data frame (silicon signature data) reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{FD2} (MAX.)$).
- <6> The received data frame (silicon signature data) is checked.

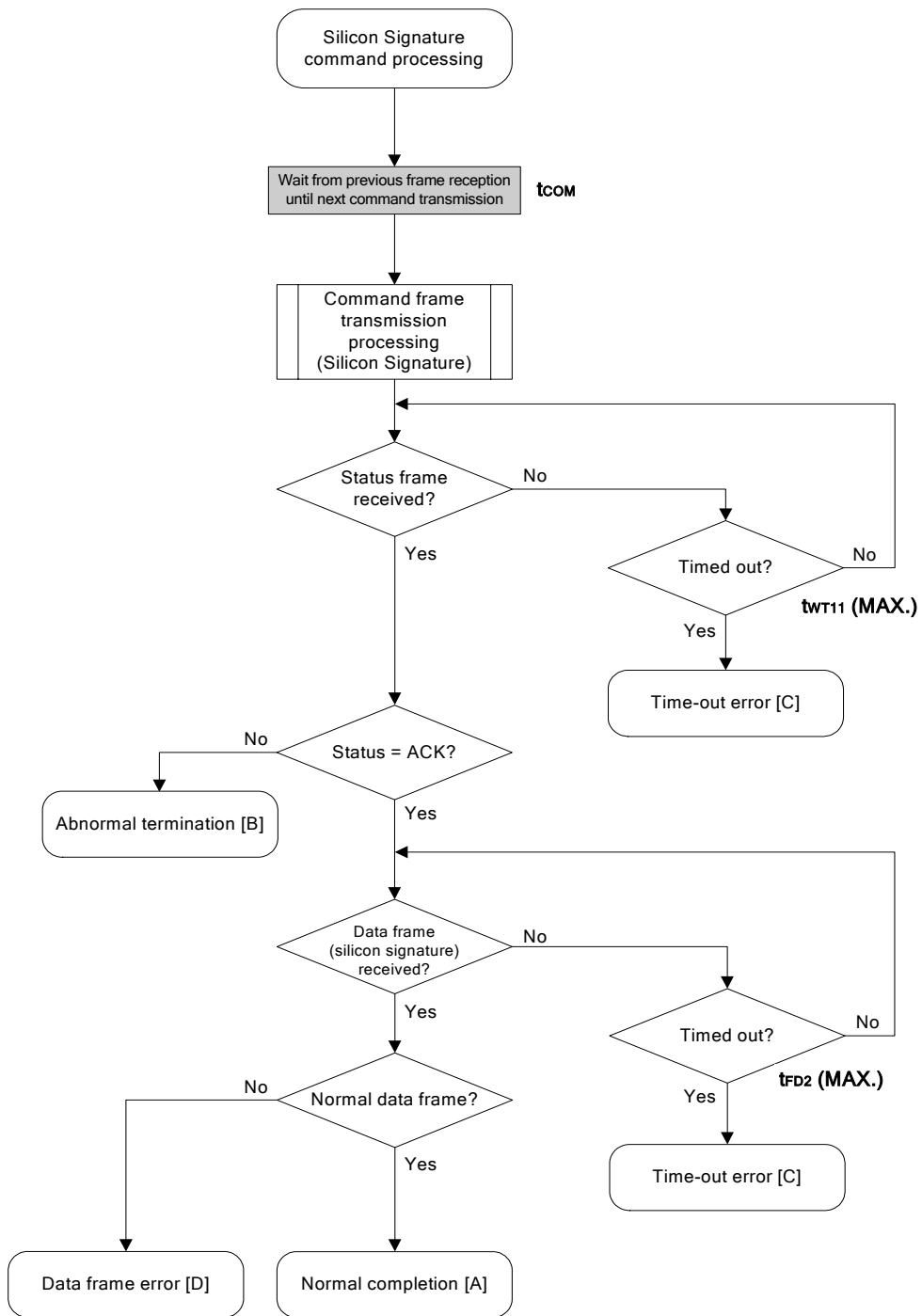
If data frame is normal: Normal completion [A]

If data frame is abnormal: Data frame error [D]

6.12.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the silicon signature was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame or data frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as silicon signature data does not match.

6.12.4 Flowchart



6.12.5 Sample program

The following shows a sample program for Silicon Signature command processing.

```

/*****
/*
/* Get silicon signature command
/*
/*****
/* [i] u8 *sig ... pointer to signature save area
/* [r] u16 ... error code
/*****
u16 fl_ua_getsig(u8 *sig)
{
    u16 rc;

    fl_wait(tCOM); // wait before sending command

    put_cmd_ua(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm); // send GET SIGNATURE command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT11_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR: break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }

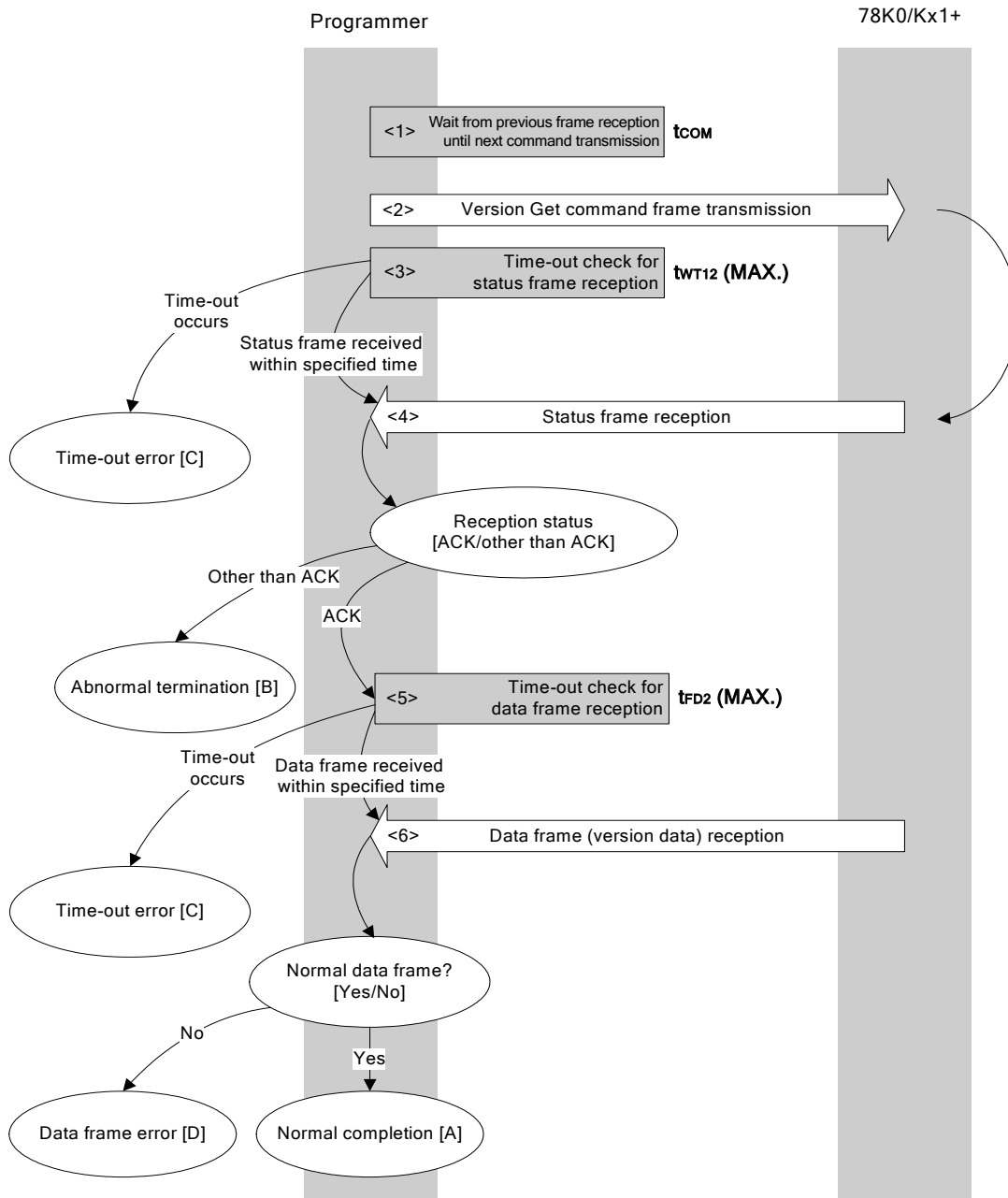
    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX); // get status frame
    if (rc){ // if error
        return rc; // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                                // copy Signature data
    return rc; // case [A]
}

```

6.13 Version Get Command

6.13.1 Processing sequence chart

Version Get command processing sequence



6.13.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Version Get command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT12} (MAX.)$).
- <4> The status code is checked.

When ST1 = ACK: Proceeds to <5>.

When ST1 \neq ACK: Abnormal termination [B]

- <5> A time-out check is performed until data frame (version data) reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{FD2} (MAX.)$).
- <6> The received data frame (version data) is checked.

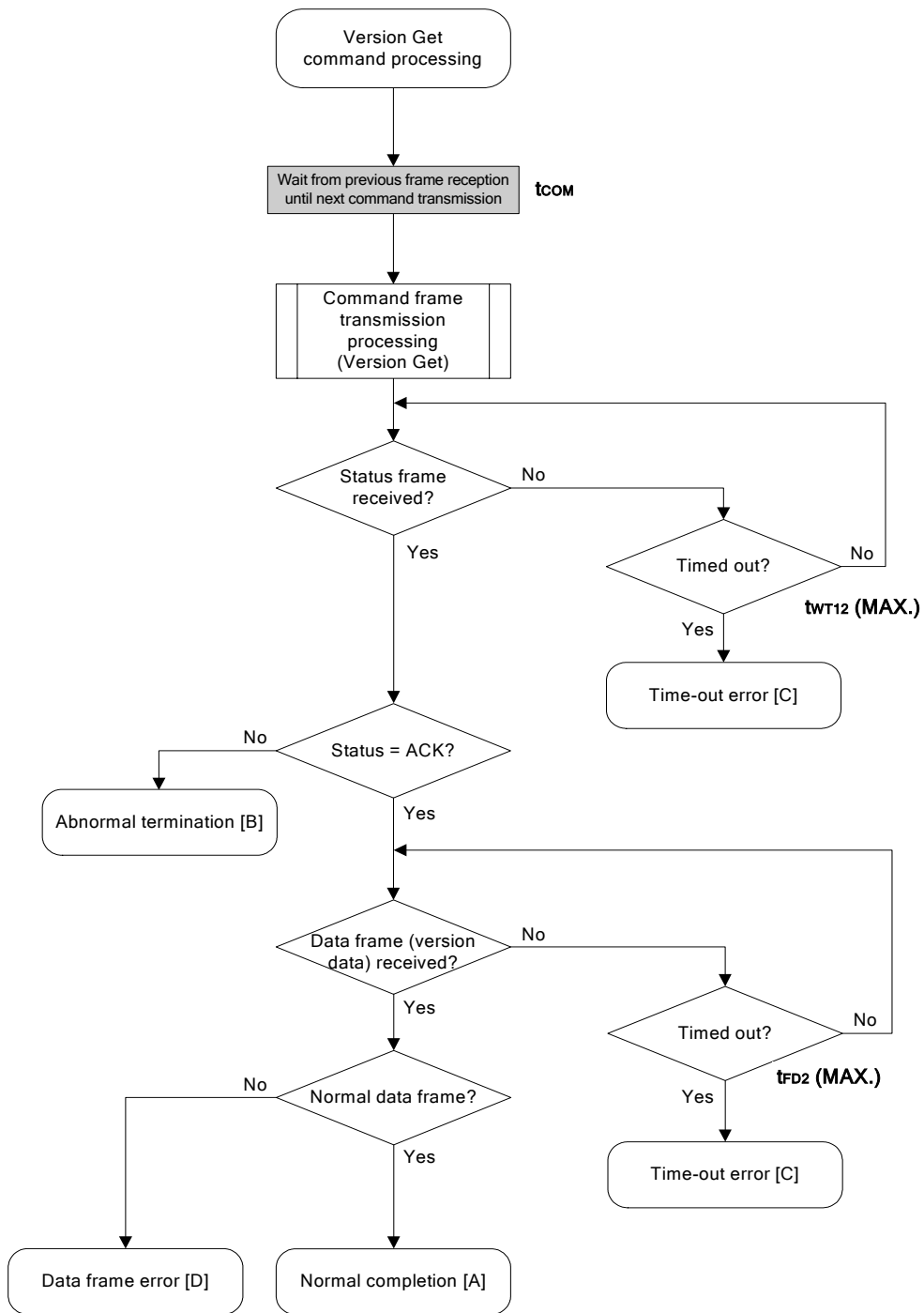
If data frame is normal: Normal completion [A]

If data frame is abnormal: Data frame error [D]

6.13.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and version data was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame or data frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as version data does not match.

6.13.4 Flowchart



6.13.5 Sample program

The following shows a sample program for Version Get command processing.

```

/*****
/*
/* Get device/firmware version command
/*
/*****
/* [i] u8 *buf ... pointer to version data save area
/* [r] u16 ... error code
/*****
u16      fl_ua_getver(u8 *buf)
{
    u16    rc;

    fl_wait(tCOM);          // wait before sending command

    put_cmd_ua(FL_COM_GET_VERSION, 1, fl_cmd_prm); // send GET VERSION command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT12_MAX);      // get status frame
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_DFTO_ERR: return rc;  break; // case [C]
        default:                  return rc;  break; // case [B]
    }

    rc = get_dfrm_ua(fl_rxdata_frm, tFD2_MAX);    // get data frame
    if (rc){
        return rc;                  // case [D]
    }

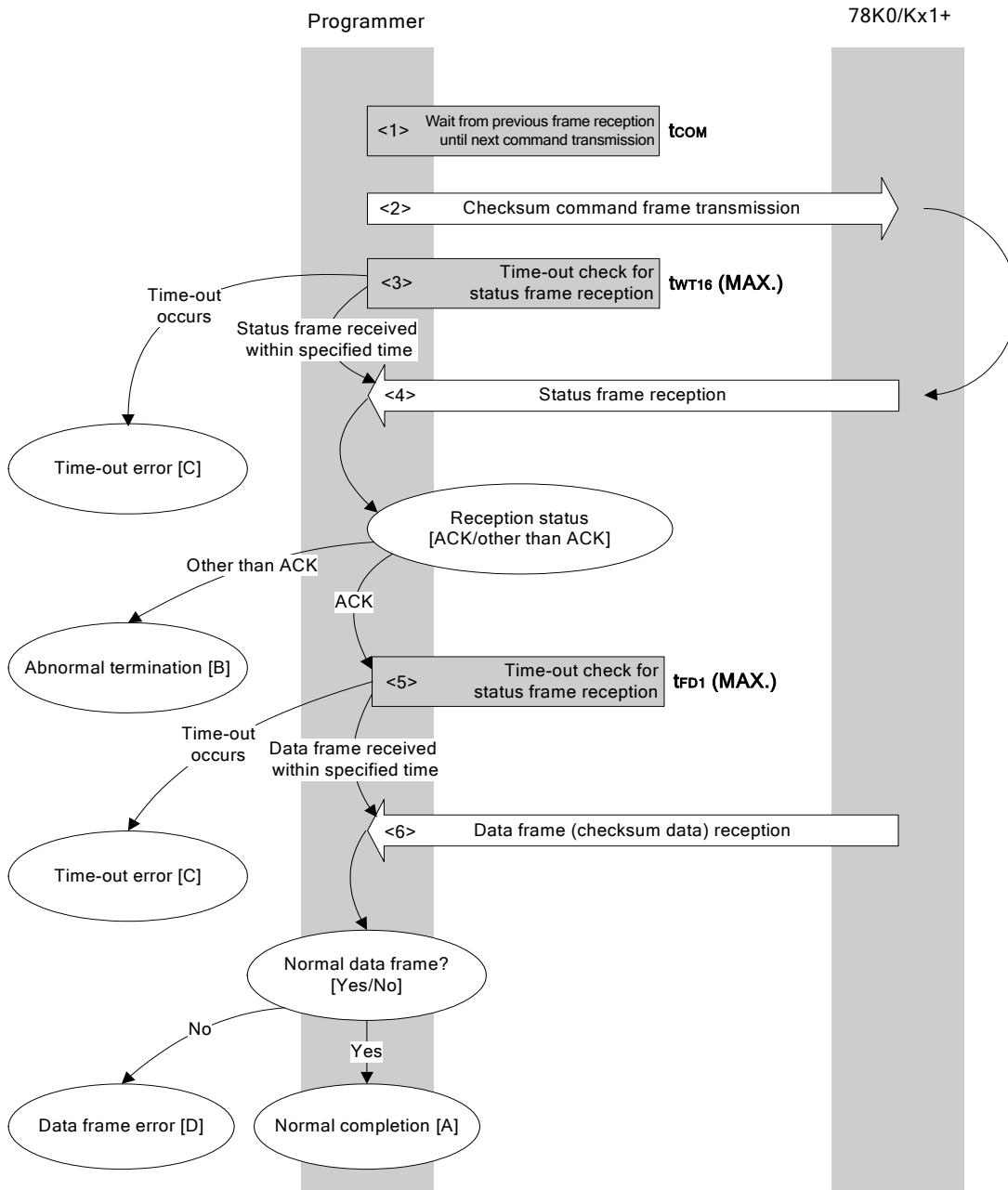
    memcpy(buf, fl_rxdata_frm+OFS_STA_PLD, DFV_LEN); // copy version data
    return rc;                      // case [A]
}

```

6.14 Checksum Command

6.14.1 Processing sequence chart

Checksum command processing sequence



6.14.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Checksum command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT16} (MAX.)$).
- <4> The status code is checked.

When ST1 = ACK: Proceeds to <5>.

When ST1 \neq ACK: Abnormal termination [B]

- <5> A time-out check is performed until data frame (checksum data) reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{FD1} (MAX.)$).
- <6> The received data frame (checksum data) is checked.

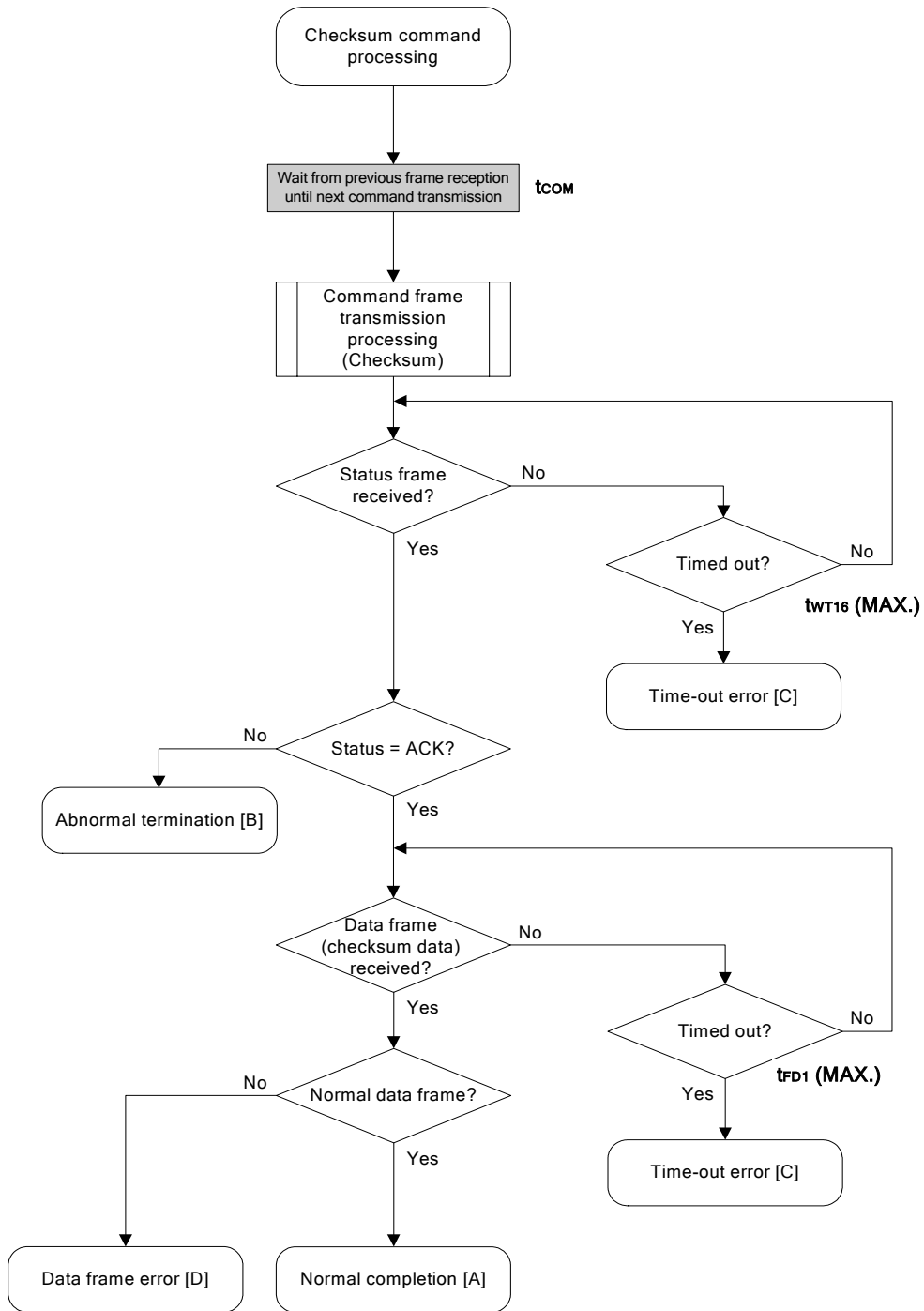
If data frame is normal: Normal completion [A]

If data frame is abnormal: Data frame error [D]

6.14.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and checksum data was acquired normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame or data frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as version data does not match.

6.14.4 Flowchart



6.14.5 Sample program

The following shows a sample program for Checksum command processing.

```

/*****/
/*
/* Get checksum command
/*
/*****/
/* [i] u16 *sum    ... pointer to checksum save area
/* [i] u32 top     ... start address
/* [i] u32 bottom  ... end address
/* [r] u16        ... error code
/*****/
u16    fl_ua_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16    rc;

    /*****/
    /*    set params
    /*****/
    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    /*****/
    /*    send command
    /*****/

    fl_wait(tCOM);          // wait before sending command

    put_cmd_ua(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // send GET VERSION command

    rc = get_sfrm_ua(fl_ua_sfrm, tWT16_MAX);          // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_DFTO_ERR: return rc;    break; // case [C]
        default:                        return rc;    break; // case [B]
    }

    /*****/
    /*    get data frame (Checksum data)
    /*****/
    rc = get_dfrm_ua(fl_rxddata_frm, tFD1_MAX);      // get status frame
    if (rc){
        return rc;                          // if no error,
        // case [D]
    }

    *sum = (fl_rxddata_frm[OFS_STA_PLD] << 8) + fl_rxddata_frm[OFS_STA_PLD+1];
        // set SUM data

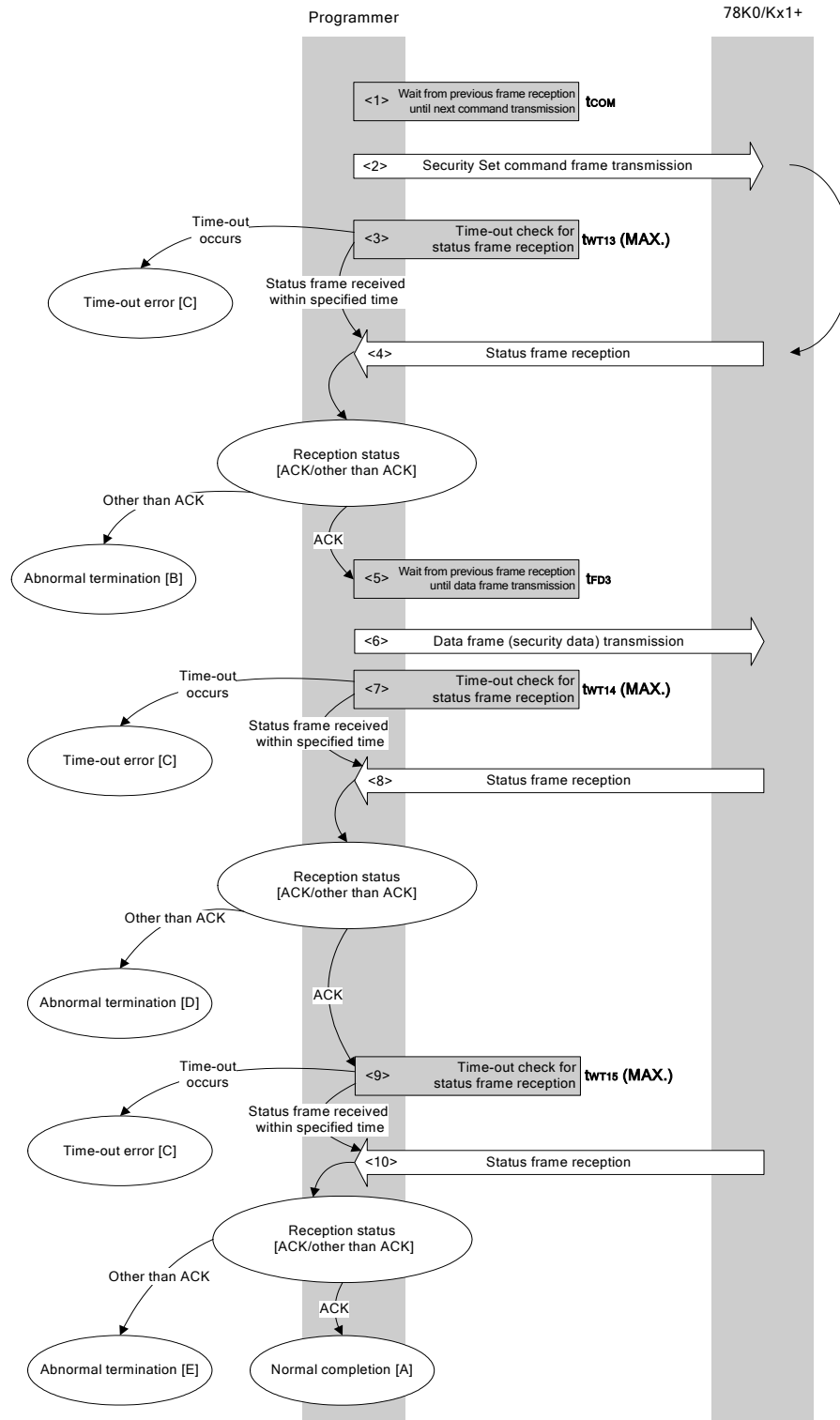
    return rc;                                // case [A]
}

```

6.15 Security Set Command

6.15.1 Processing sequence chart

Security Set command processing sequence



6.15.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Security Set command is transmitted by command frame transmission processing.
- <3> A time-out check is performed from command transmission until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time t_{WT13} (MAX.)).
- <4> The status code is checked.

When ST1 = ACK: Proceeds to <5>.

When ST1 \neq ACK: Abnormal termination [B]

- <5> Waits from the previous frame reception until the next data frame transmission (wait time t_{FD3}).
- <6> The data frame (security setting data) is transmitted by data frame transmission processing.
- <7> A time-out check is performed until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time t_{WT14} (MAX.)).
- <8> The status code is checked.

When ST1 = ACK: Proceeds to <9>.

When ST1 \neq ACK: Abnormal termination [D]

- <9> A time-out check is performed until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time t_{WT15} (MAX.)).
- <10> The status code is checked.

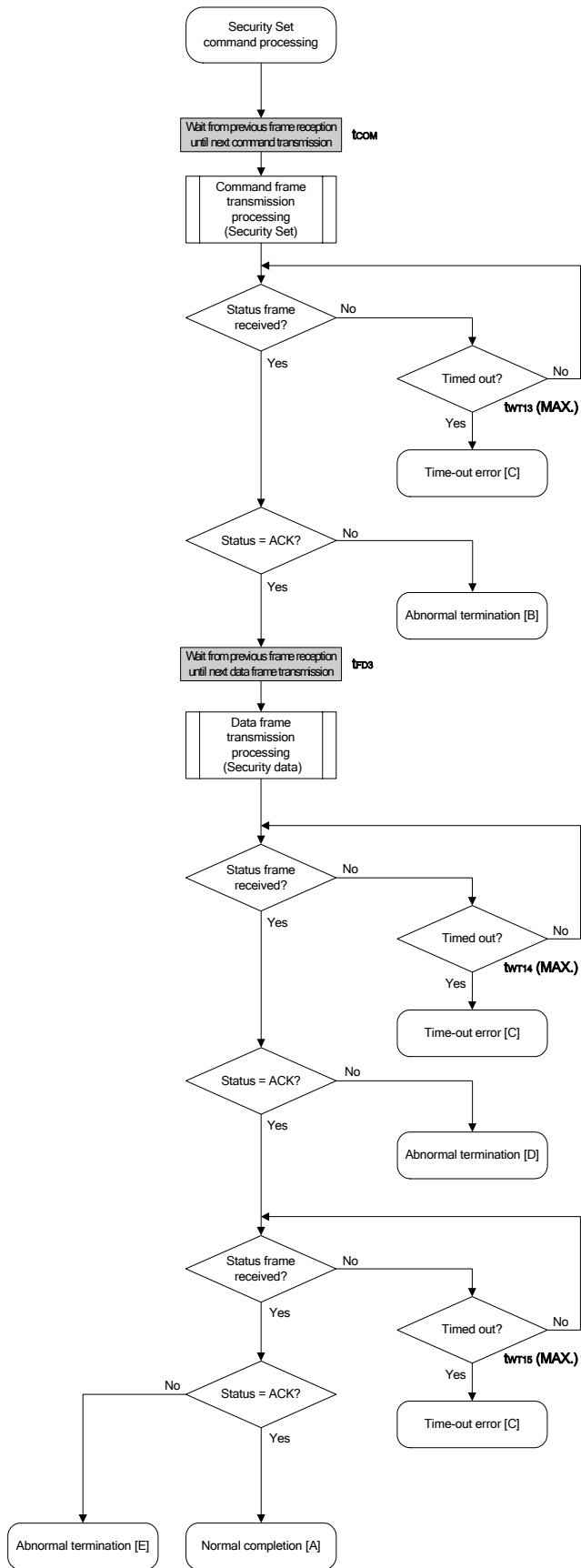
When ST1 = ACK: Normal completion [A]

When ST1 \neq ACK: Abnormal termination [E]

6.15.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and security setting was performed normally.
Abnormal termination [B]	Parameter error	05H	BLK or PAG is not 00H.
	Checksum error	07H	The checksum of the transmitted command frame or data frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame or data frame was not received within the specified time.
Abnormal termination [D]	FLMD error	18H	A write error has occurred.
	Write error	1CH	A write error has occurred (including the case of security data already being set).
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

6.15.4 Flowchart



6.15.5 Sample program

The following shows a sample program for Security Set command processing.

```

/*****/
/*
/* Set security flag command
/*
/*****/
/* [i] u8 scf ... Security flag data
/* [r] u16 ... error code
/*****/
u16 fl_ua_setscf(u8 scf)
{
    u16 rc;

    /*****/
    /* set params
    /*
    /*****/
    fl_cmd_prm[0] = 0x00; // "BLK" (must be 0x00)
    fl_cmd_prm[1] = 0x00; // "PAG" (must be 0x00)
    fl_txdata_frm[0] = scf|= 0b11111000;
                                // "FLG" (upper 5bits must be '1' (to make sure))

    /*****/
    /* send command
    /*
    /*****/
    fl_wait(tCOM); // wait before sending command

    put_cmd_ua(FL_COM_SET_SECURITY, 3, fl_cmd_prm);

    rc = get_sfrm_ua(fl_ua_sfrm, tWT13_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR: break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }

    /*****/
    /* send data frame (security setting data)
    /*
    /*****/

    fl_wait(tFD3);
    put_dfrm_ua(1, fl_txdata_frm, true); // send securithi setting data

    rc = get_sfrm_ua(fl_ua_sfrm, tWT14_MAX); // get status frame
    switch(rc) {
        case FLC_NO_ERR: break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default: return rc; break; // case [B]
    }

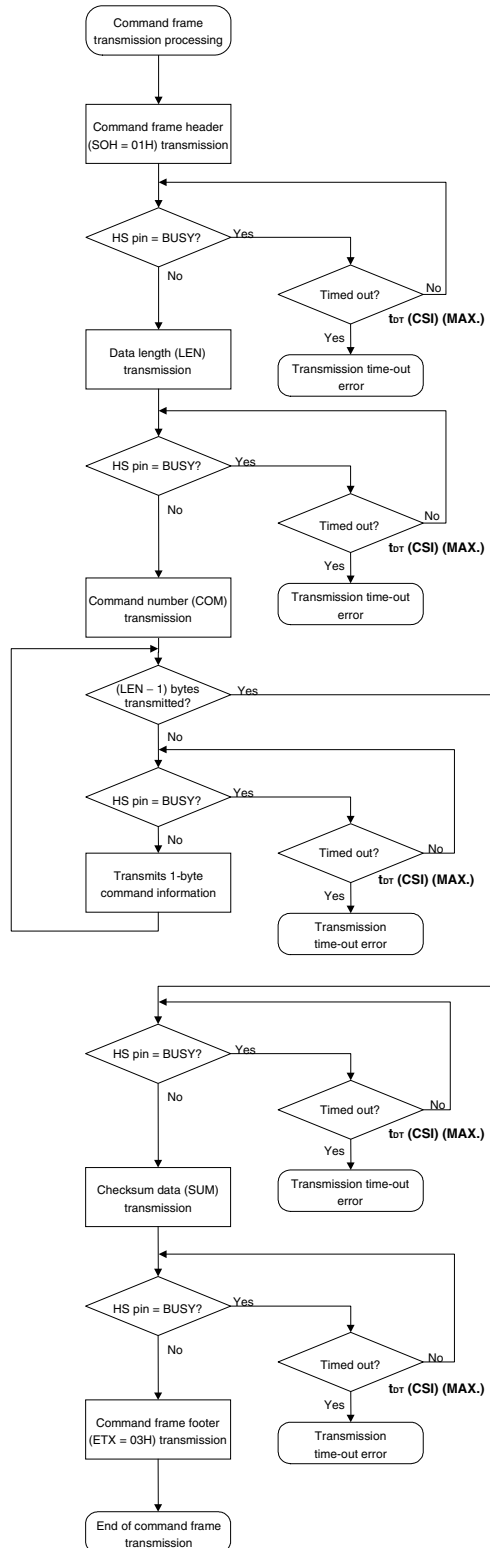
    /*****/
    /* Check internally verify
    /*
    /*****/

```

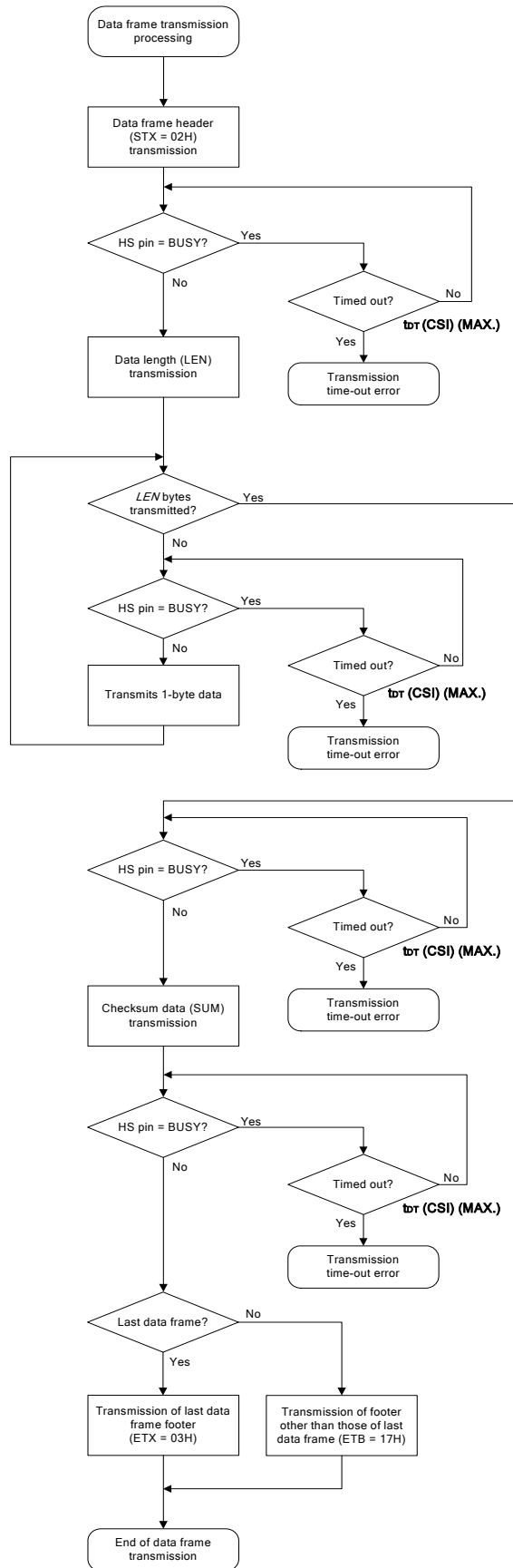
```
rc = get_sfrm_ua(fl_ua_sfrm, tWT15_MAX);      // get status frame
// switch(rc) {
//
//     case FLC_NO_ERR: return rc; break; // case [A]
//     case FLC_DFTO_ERR: return rc; break; // case [C]
//     default: return rc; break; // case [B]
// }
return rc;
}
```

CHAPTER 7 3-WIRE SERIAL I/O COMMUNICATION MODE WITH HANDSHAKE SUPPORTED (CSI + HS)

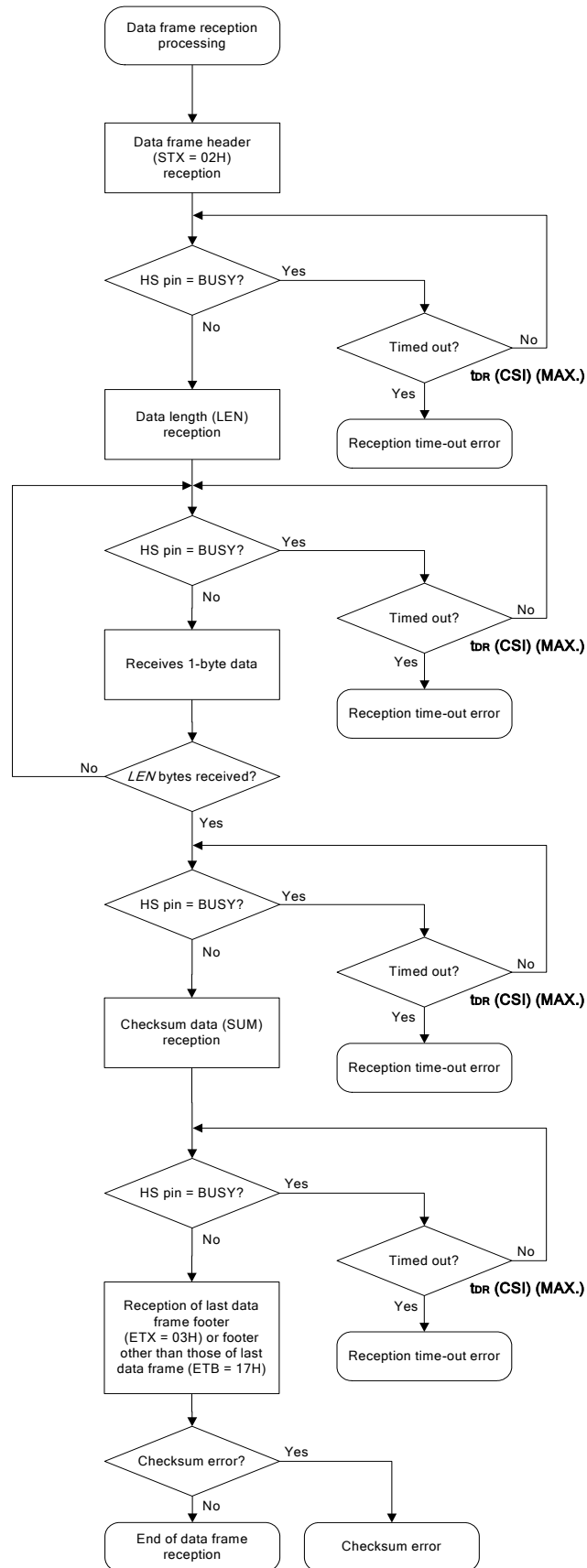
7.1 Command Frame Transmission Processing Flowchart



7.2 Data Frame Transmission Processing Flowchart

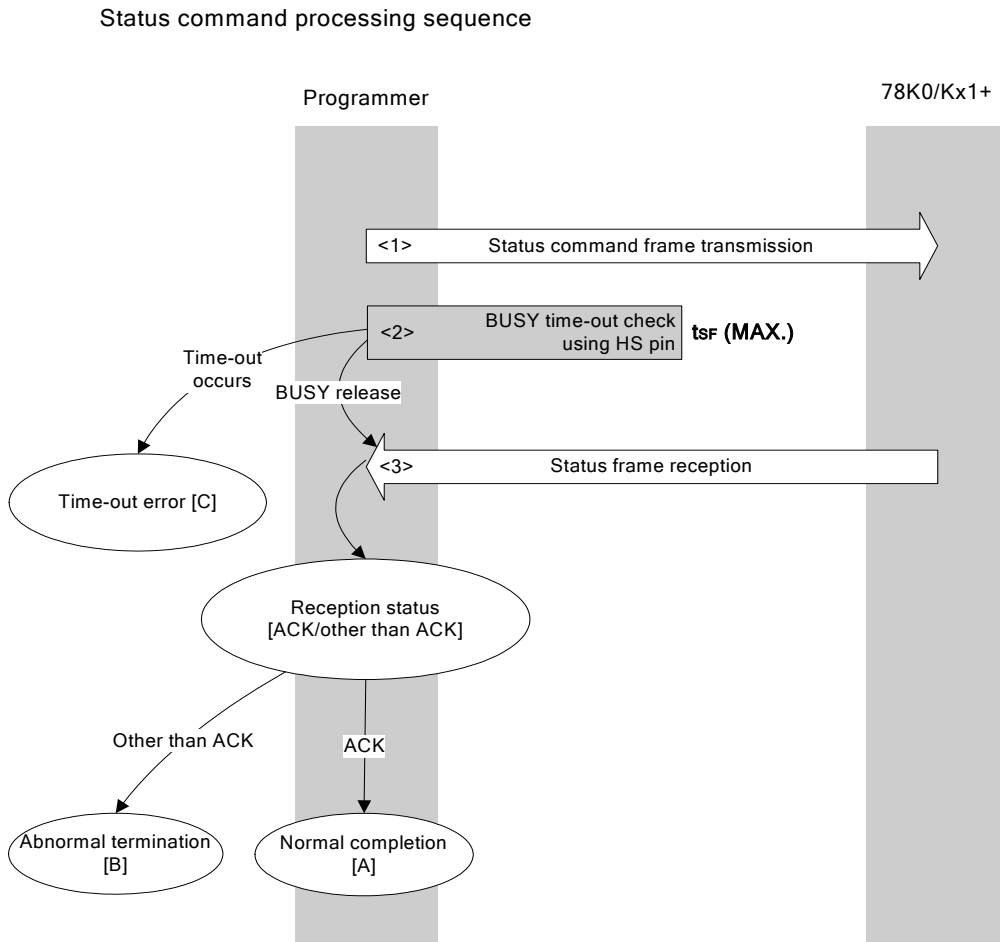


7.3 Data Frame Reception Processing Flowchart



7.4 Status Command

7.4.1 Processing sequence chart



7.4.2 Description of processing sequence

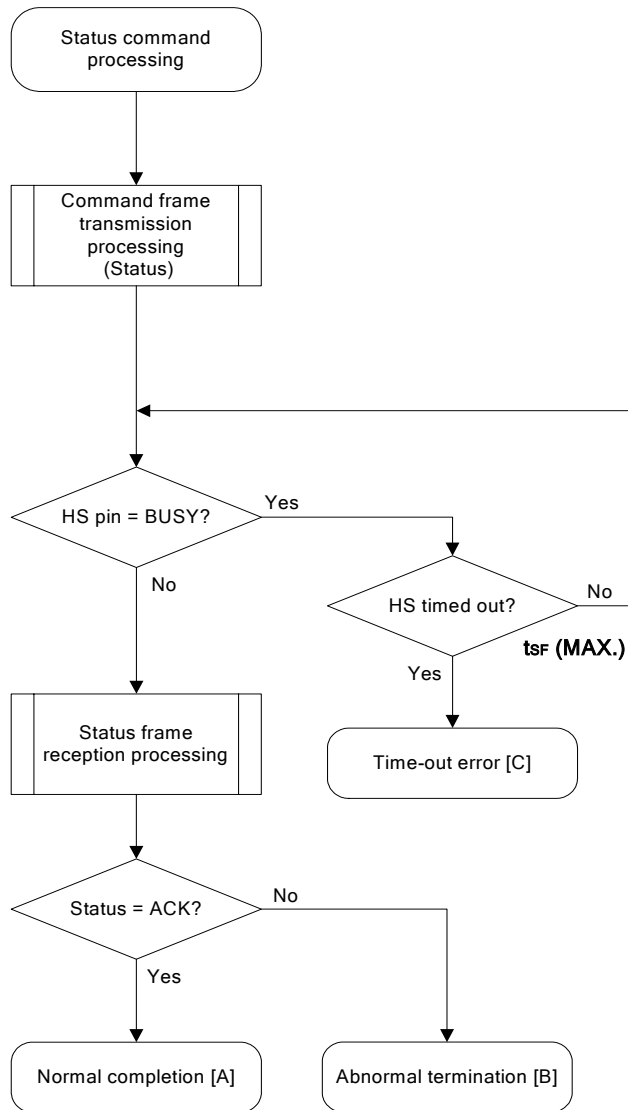
- <1> The Status command is transmitted by command frame transmission processing.
- <2> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{SF(MAX)}$).
- <3> The status code is checked.

When ST1 = ACK: Normal completion [A]
 When ST1 ≠ ACK: Abnormal termination [B]

7.4.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The status frame transmitted from the 78K0/Kx1+ has been received normally.
Abnormal termination [B]	Command error	04H	An unsupported command or abnormal frame has been received.
	Parameter error	05H	Command information (parameter) is invalid.
	Checksum error	07H	The data of the frame transmitted from the programmer is abnormal.
	Verify error	0FH	A verify error has occurred for the data of the frame transmitted from the programmer.
	Protect error	10H	An attempt was made to execute processing prohibited by the Security Set command.
	Negative acknowledgment (NACK)	15H	Negative acknowledgment
	FLMD error	18H	A write error has occurred.
	MRG10 error	1AH	An erase error has occurred.
	MRG11 error	1BH	An internal verify error has occurred during data write, or a blank check error has occurred.
	Write error	1CH	A write error has occurred.
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.

7.4.4 Flowchart



7.4.5 Sample program

The following shows a sample program for Status command processing.

```

/*****
/*
/* Get status command (CSI-HS)
/*
/*****
/* [r] u16      ... decoded status or error code
/*
/* (see fl.h/fl-proto.h &
/*      definition of decode_status() in fl.c)
/*****
static u16 fl_hs_getstatus(void)
{
    u16    rc;
    u32    retry = 0;

    rc = put_cmd_hs(FL_COM_GET_STA, 1, fl_cmd_prm);    // send "Status" command
    if (rc)
        return rc;    // case [C]

    if (hs_busy_to(tSF_MAX))    // HS-Busy t.o. ?
        return FLC_HSTO_ERR;    // t.o. detected : case [C]

    if (rc = get_sfrm_hs(fl_rxdata_frm))
        return rc;    // case [C] or [B(checksum error)]

    rc = decode_status(fl_st1);    // decode return code

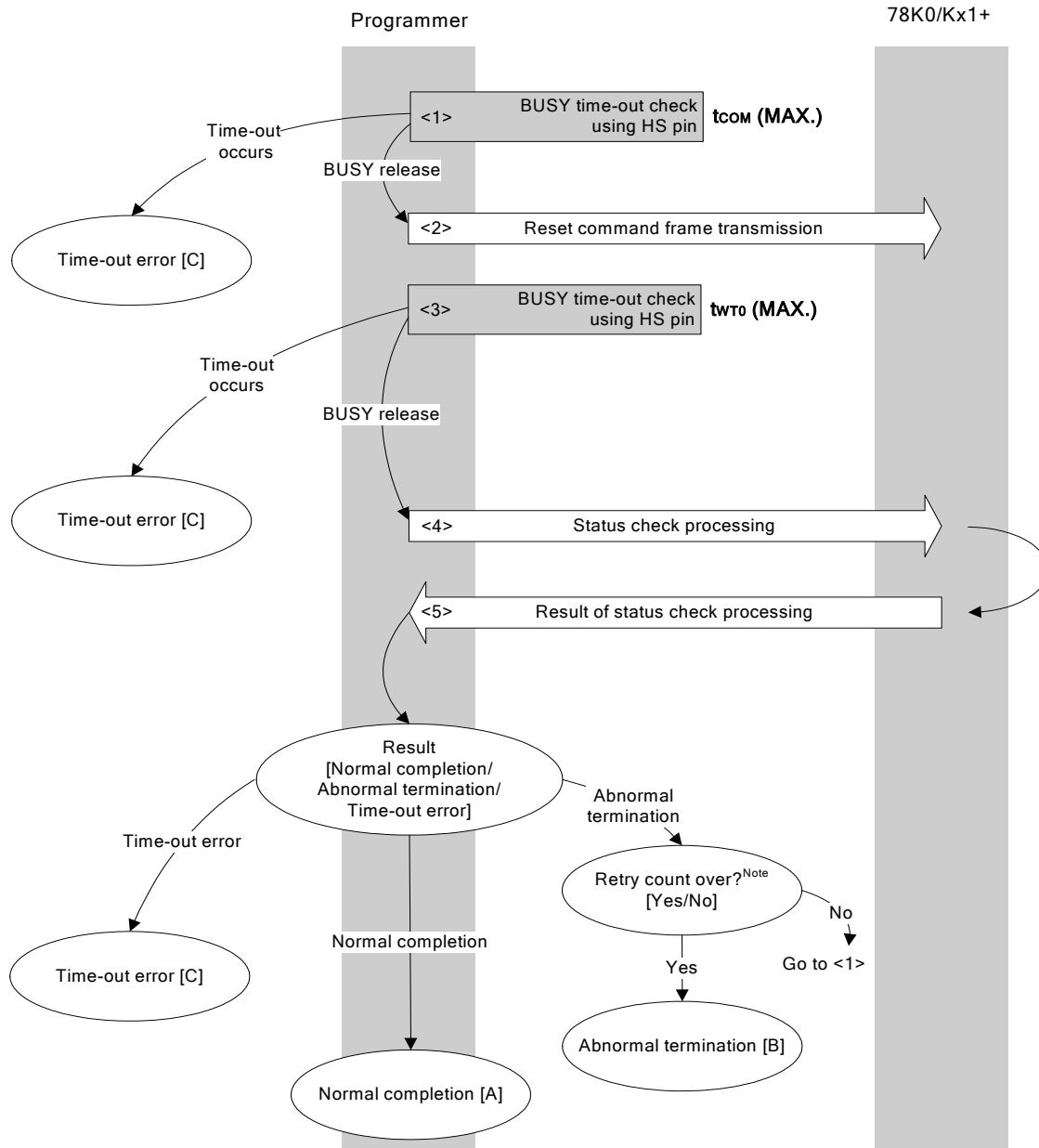
    return rc;    // case [A] or [B]
}

```

7.5 Reset Command

7.5.1 Processing sequence chart

Reset command processing sequence



Note Do not exceed the retry count for the reset command transmission (up to 16 times).

7.5.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM(MAX)}$).
- <2> The Reset command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WTO(MAX)}$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]

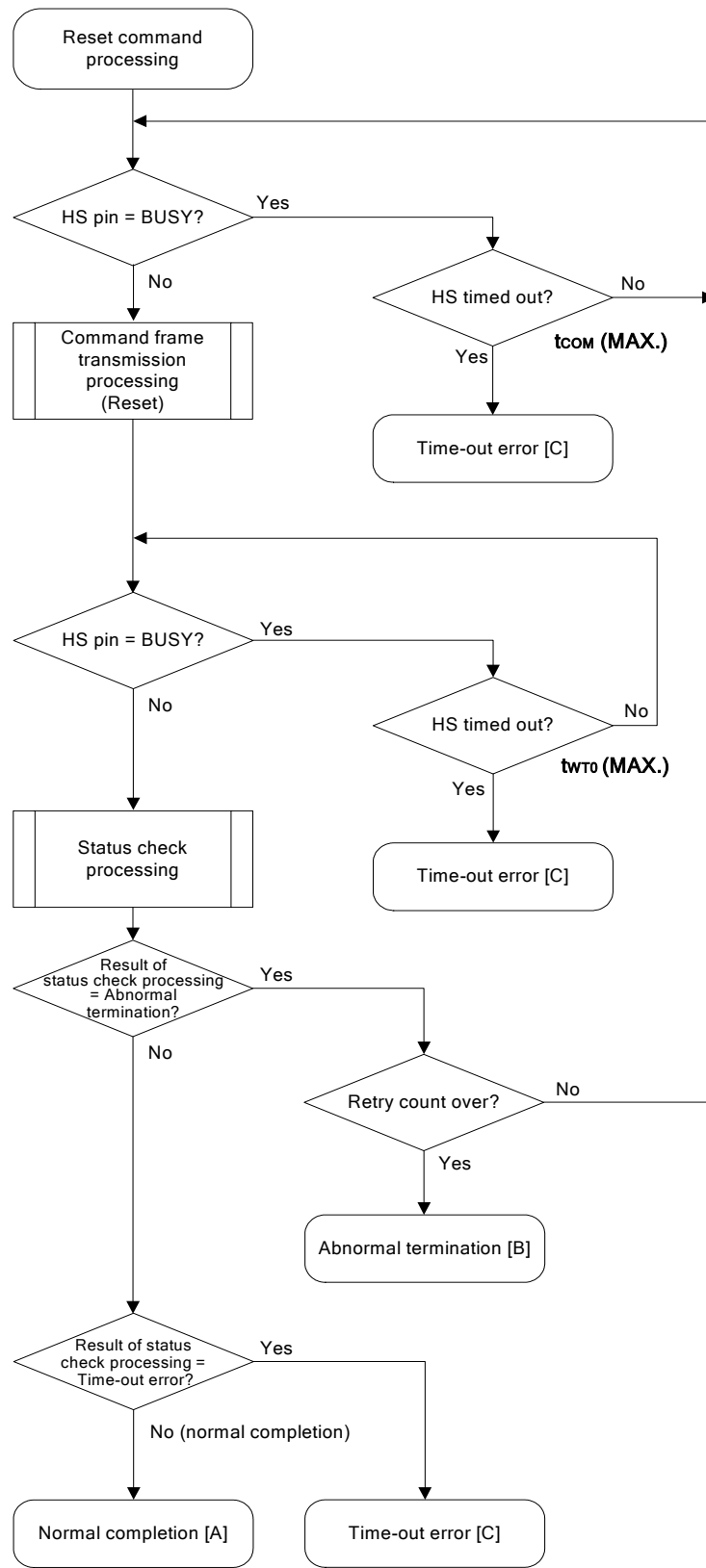
When the processing ends abnormally: The sequence is re-executed from <1> if the retry count is not over.
If the retry count is over, the processing ends abnormally [B].

When a time-out error occurs: A time-out error [C] is returned.

7.5.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and synchronization between the programmer and the 78K0/Kx1+ has been established.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Status check processing timed out. Processing timed out due to the busy status at the HS pin.

7.5.4 Flowchart



7.5.5 Sample program

The following shows a sample program for Reset command processing.

```

/*****
/*
/* Reset command (CSI-HS)
/*
/*
/*****
/* [r] u16          ... error code
/*****
u16          fl_hs_reset(void)
{
    u16      rc;
    u32      retry;

    for (retry = 0; retry < tRS; retry++){

        if (hs_busy_to(tCOM_MAX))
            return FLC_HSTO_ERR;          // t.o. detected :case [C]

        rc = put_cmd_hs(FL_COM_RESET, 1, fl_cmd_prm); // send "Reset" command
        if (rc)
            return rc;          // case [C]

        if (hs_busy_to(tWT0_MAX))
            return FLC_HSTO_ERR;          // t.o. detected :case [C]

        rc = fl_hs_getstatus(); // get status frame
        if (rc == FLC_ACK)      // ST1 = ACK ?
            break;              // case [A]
        //continue;            // case [B] (if exit from loop)

    }

    // switch(rc) {
    //     case  FLC_NO_ERR:  return rc;   break; // case [A]
    //     case  FLC_HSTO_ERR: return rc;  break; // case [C]
    //     default:         return rc;   break; // case [B]
    // }

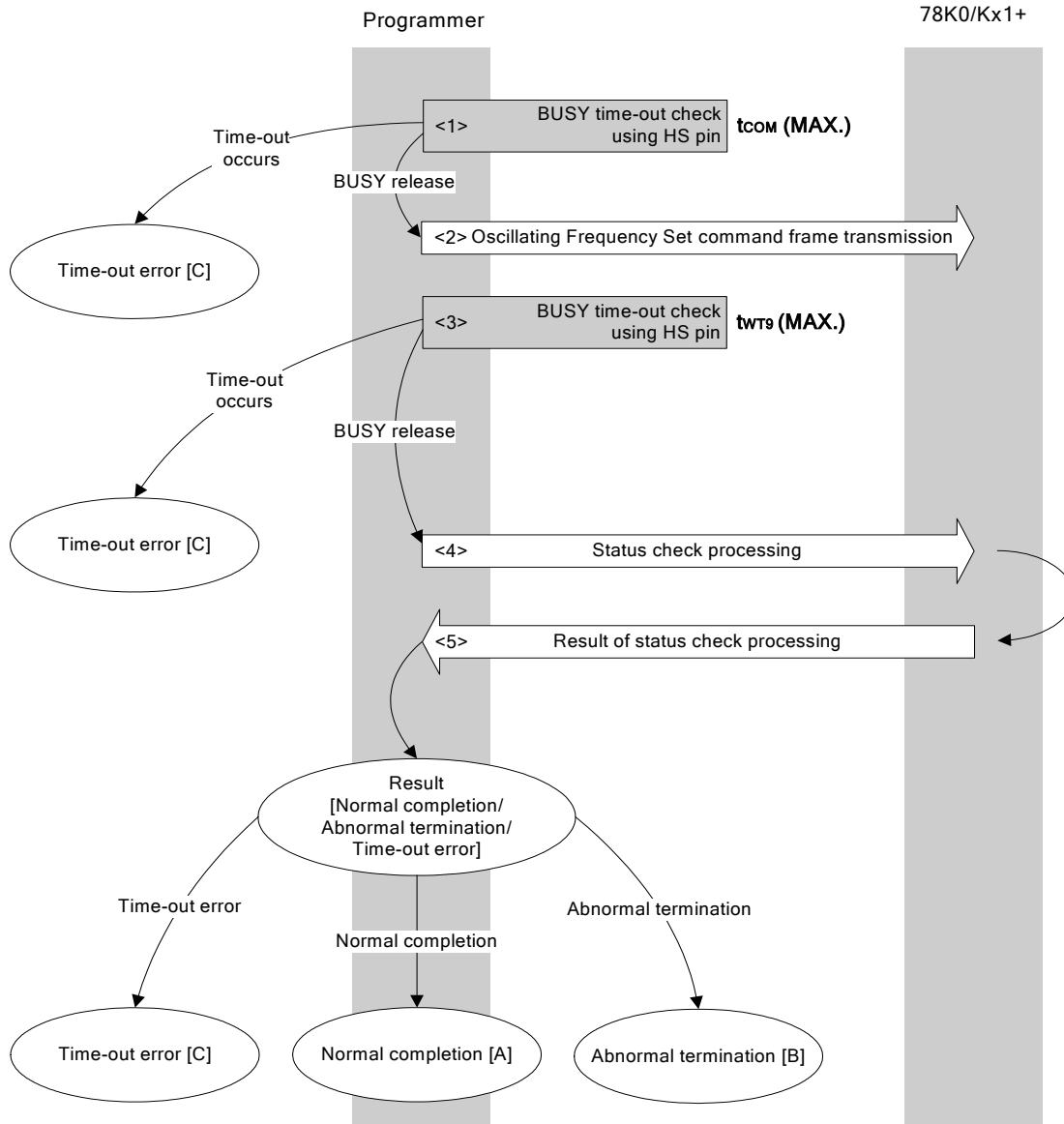
    return rc;
}

```

7.6 Oscillating Frequency Set Command

7.6.1 Processing sequence chart

Oscillating Frequency Set command processing sequence



7.6.2 Description of processing sequence

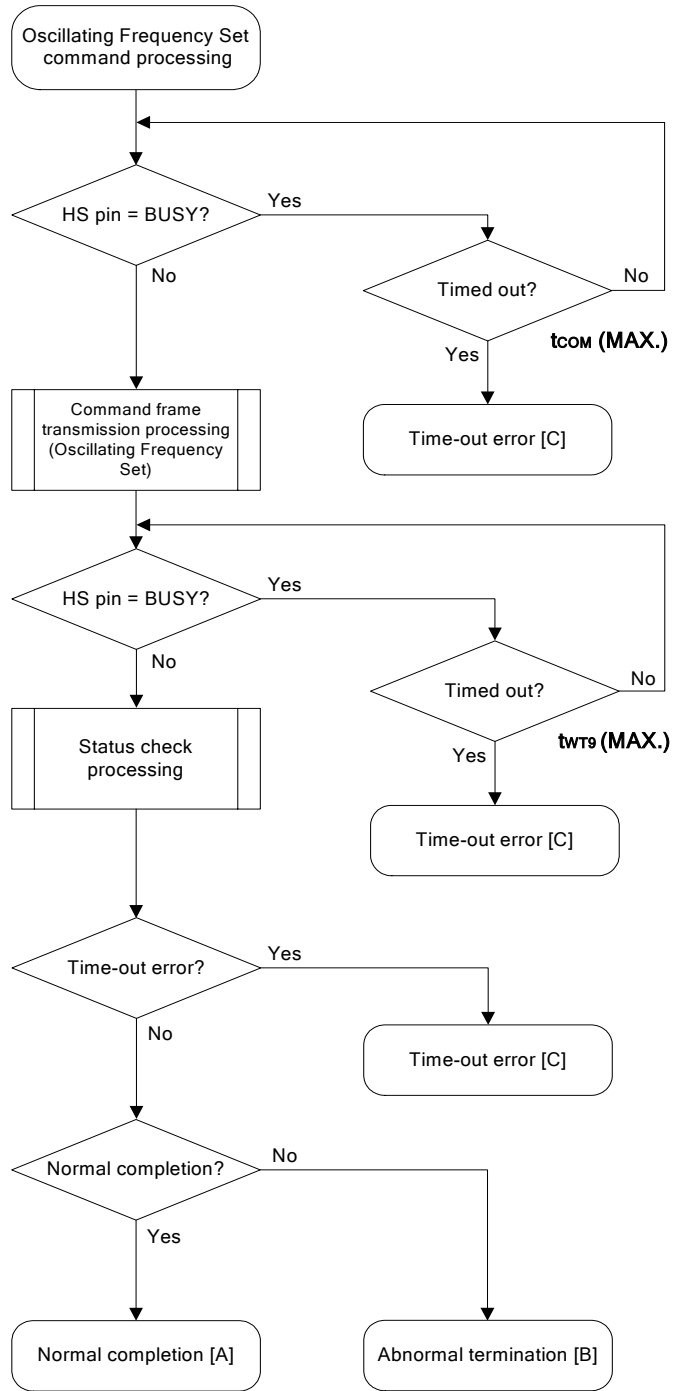
- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM}(MAX.)$).
- <2> The Oscillating Frequency Set command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT9}(MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

7.6.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the operating frequency was correctly set to the 78K0/Kx1+.
Abnormal termination [B]	Parameter error	05H	The oscillation frequency value is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.

7.6.4 Flowchart



7.6.5 Sample program

The following shows a sample program for Oscillating Frequency Set command processing.

```

/*****
/*
/* Set Flash device clock value command (CSI-HS)
/*
/*
/*****
/* [i] u8 clk[4] ... frequency data(D1-D4)
/* [r] u16 ... error code
/*****
/*****
u16 fl_hs_setclk(u8 clk[])
{
    u16 rc;

    fl_cmd_prm[0] = clk[0]; // "D01"
    fl_cmd_prm[1] = clk[1]; // "D02"
    fl_cmd_prm[2] = clk[2]; // "D03"
    fl_cmd_prm[3] = clk[3]; // "D04"

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm))
        // send "Oscilating Frequency Set" command
        return rc; // case [C]

    if (hs_busy_to(tWT9_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

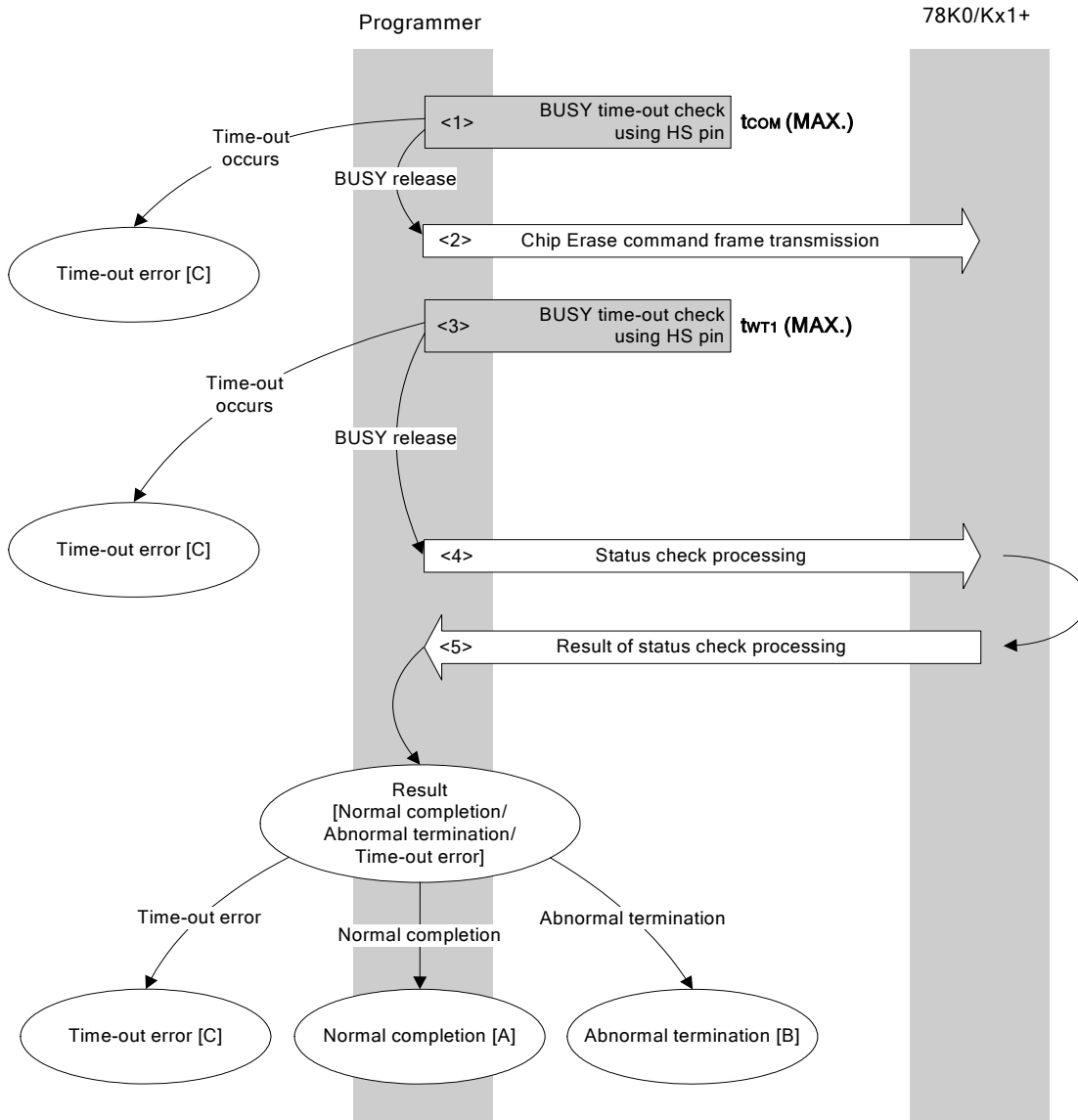
    rc = fl_hs_getstatus(); // get status frame
    // switch(rc) {
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_HSTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}

```

7.7 Chip Erase Command

7.7.1 Processing sequence chart

Chip Erase command processing sequence



7.7.2 Description of processing sequence

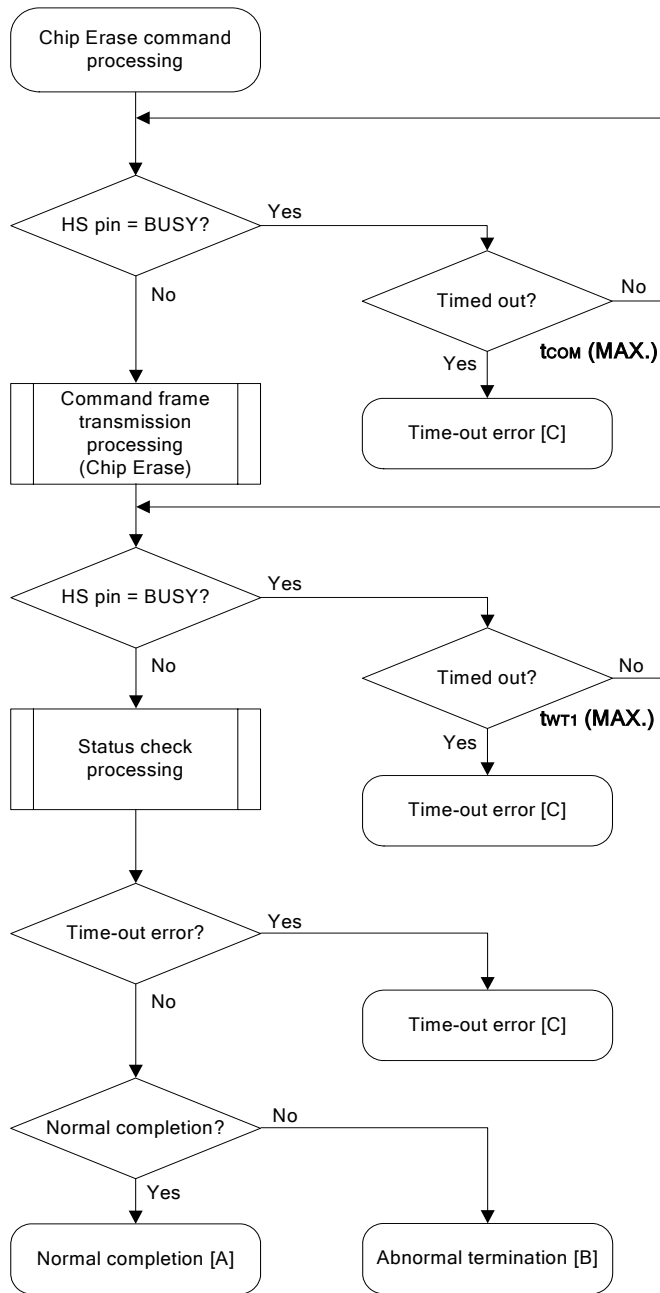
- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM(MAX)}$).
- <2> The Chip Erase command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT1(MAX)}$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

7.7.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and chip erase was performed normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Chip erase is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Erase error	1AH	An erase error has occurred.
Time-out error [C]		-	Processing timed out due to the busy status at the HS pin.

7.7.4 Flowchart



7.7.5 Sample program

The following shows a sample program for Chip Erase command processing.

```

/*****
/*
/* Erase all(chip) command (CSI-HS)
/*
/*
/*****
/* [r] u16          ... error code
/*****
u16          fl_hs_erase_all(void)
{
    u16      rc;

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;          // t.o. detected

    if (rc = put_cmd_hs(FL_COM_ERASE_CHIP, 1, fl_cmd_prm))
        return rc;                    // send "Chip Erase" command
        // case [C]

    if (hs_busy_to(tWT1_MAX))
        return FLC_HSTO_ERR;          // case [C]

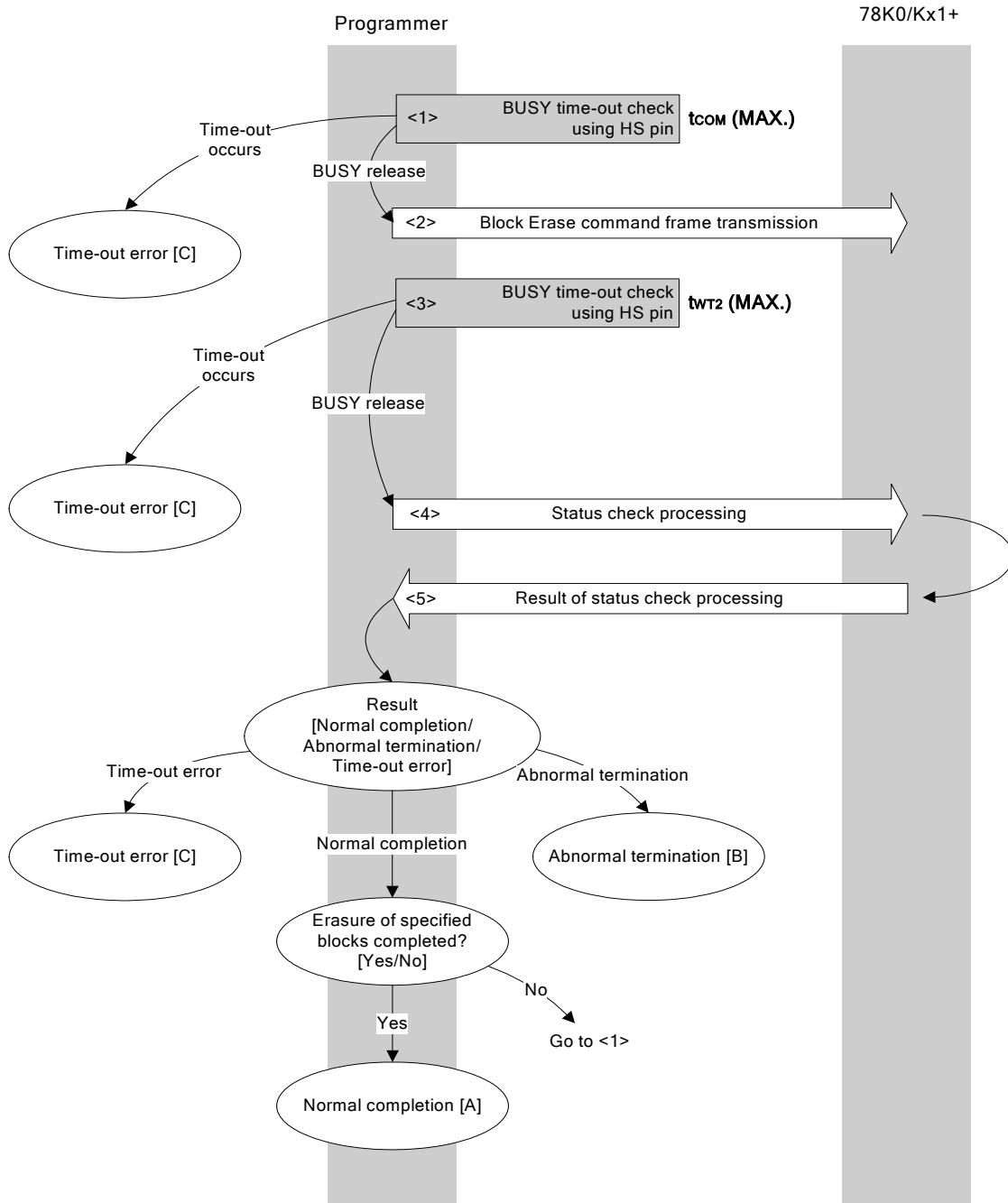
    rc = fl_hs_getstatus();            // get status frame
    // switch(rc) {
    //     case  FLC_NO_ERR:  return rc;  break; // case [A]
    //     case  FLC_HSTO_ERR: return rc;  break; // case [C]
    //     default:         return rc;   break; // case [B]
    // }
    return rc;
}

```

7.8 Block Erase Command

7.8.1 Processing sequence chart

Block Erase command processing sequence



7.8.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM}(MAX.)$).
- <2> The Block Erase command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT2}(MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: When the block erase for all of the specified blocks is not yet completed, processing changes the block number and re-executes the sequence from <1>.

When the block erase for all of the specified blocks is completed, the processing ends normally [A].

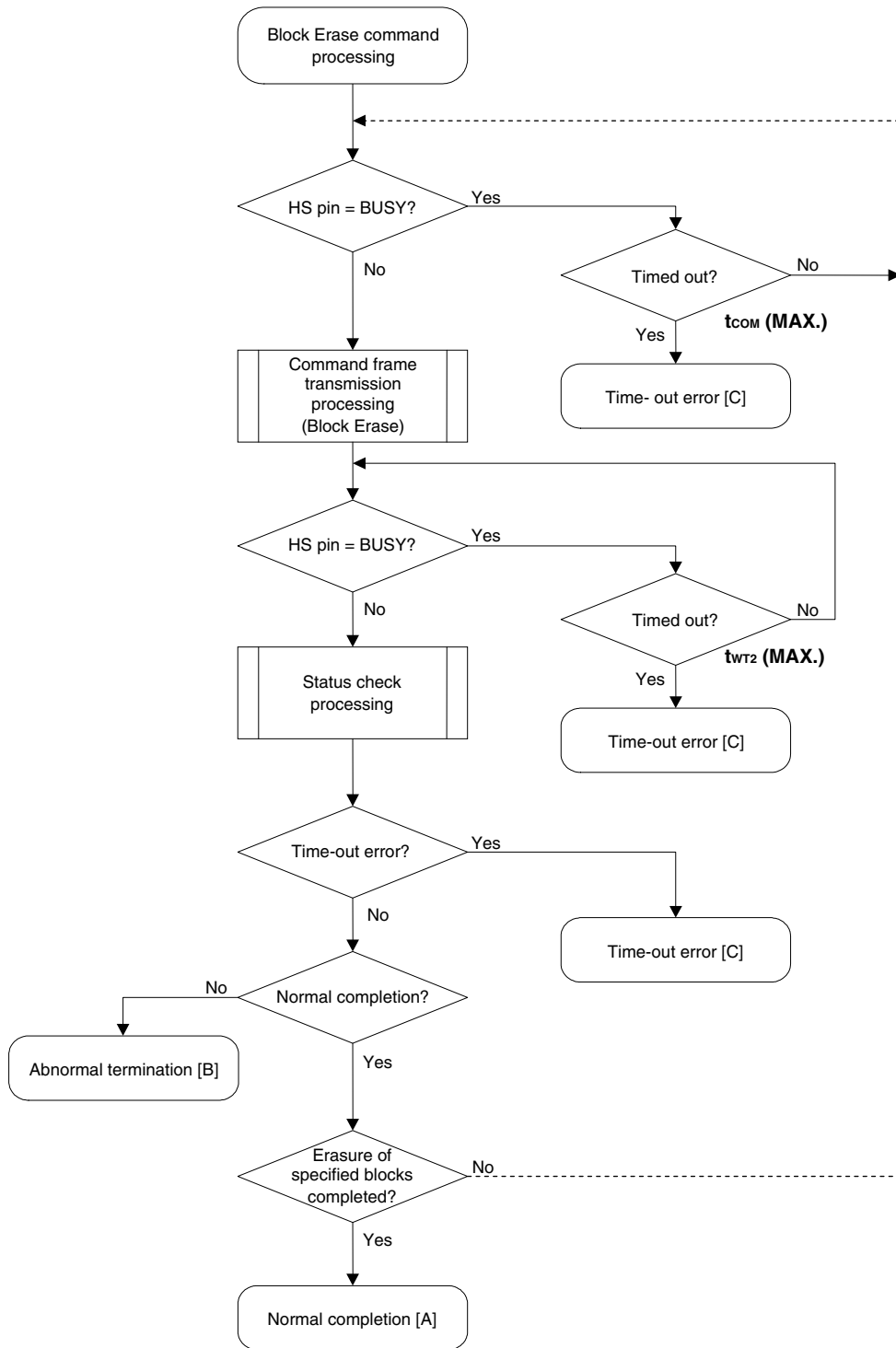
When the processing ends abnormally: Abnormal termination [B]

When a time-out error occurs: A time-out error [C] is returned.

7.8.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and block erase was performed normally.
Abnormal termination [B]	Parameter error	05H	The block number is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Chip erase is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Erase error	1AH	An erase error has occurred.
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.

7.8.4 Flowchart



7.8.5 Sample program

The following shows a sample program for Block Erase command processing for one block.

```

/*****
/*
/* Erase block command (CSI-HS)
/*
/*
/*****
/* [i] u8 block    ... block number
/* [r] u16         ... error code
/*****
u16    fl_hs_erase_blk(u8 block)
{
    u16    rc;

    fl_cmd_prm[0] = block;          // set param (BLK)

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm))
        // send "Block Erase" command
        return rc;                // case [C]

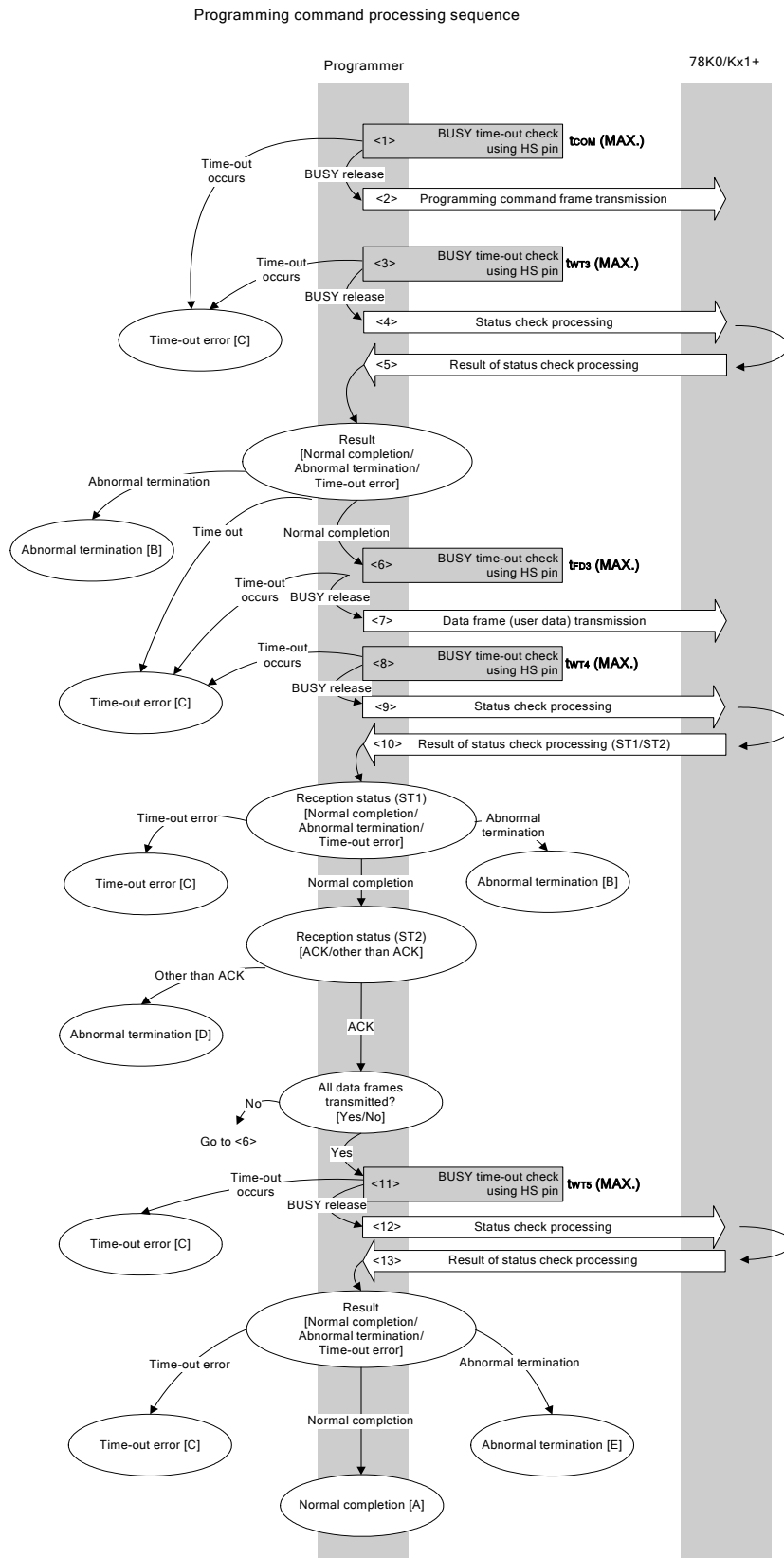
    if (hs_busy_to(tWT2_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    rc = fl_hs_getstatus();        // get status frame
// switch(rc) {
//     case  FLC_NO_ERR:  return rc;  break; // case [A]
//     case  FLC_HSTO_ERR: return rc;  break; // case [C]
//     default:         return rc;   break; // case [B]
// }
    return rc;
}

```

7.9 Programming Command

7.9.1 Processing sequence chart



7.9.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM}(MAX.)$).
- <2> The Programming command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT3}(MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
When the processing ends abnormally: Abnormal termination [B]
When a time-out error occurs: A time-out error [C] is returned.

- <6> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD3}(MAX.)$).
- <7> User data is transmitted by data frame transmission processing.
- <8> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT4}(MAX.)$).
- <9> The status frame is acquired by status check processing.
- <10> The following processing is performed according to the result of status check processing (status code (ST1/ST2)) (also refer to the processing sequence chart and flowchart).

When ST1 = abnormal termination: Abnormal termination [B]
When ST1 = time-out error: A time-out error [C] is returned.
When ST1 = normal completion: The following processing is performed according to the ST2 value.

- When ST2 ≠ ACK: Abnormal termination [D]
- When ST2 = ACK: Proceeds to <11> when transmission of all of the user data is completed.
If there still remain user data to be transmitted, the processing re-executes the sequence from <6>.

- <11> A time-out check is performed until status frame reception.
If a time-out occurs, a time-out error [C] is returned (time-out time $t_{WT5}(MAX.)$).
- <12> The status frame is acquired by status check processing.
- <13> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
(Indicating that the internal verify check has performed normally after completion of write)

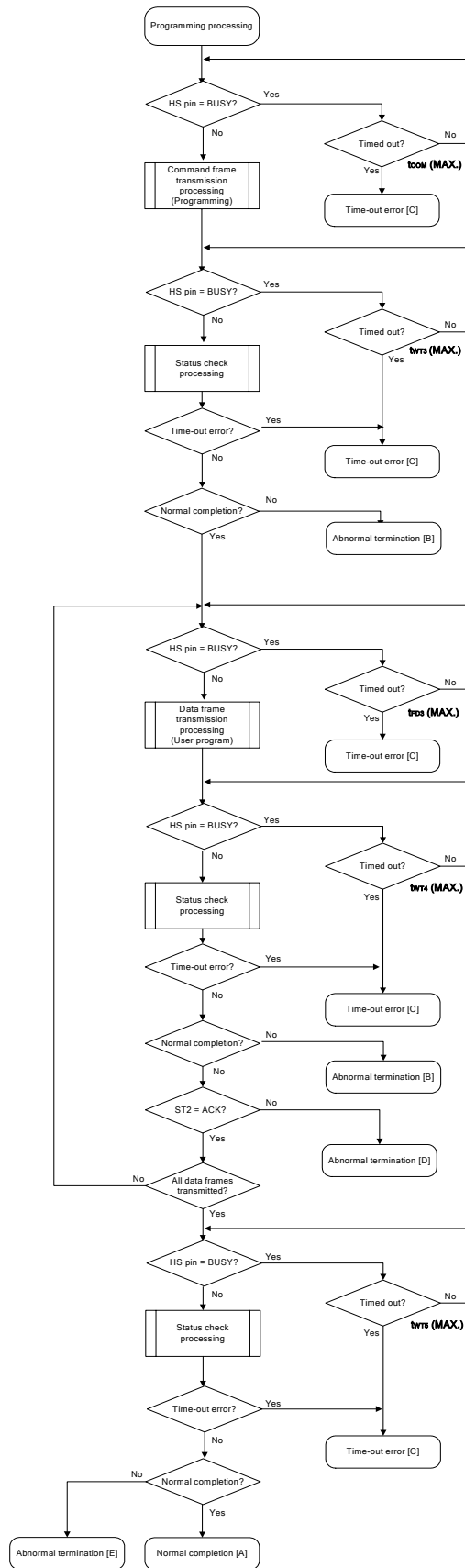
When the processing ends abnormally: Abnormal termination [E]
(Indicating that the internal verify check has not performed normally after completion of write)

When a time-out error occurs: A time-out error [C] is returned.

7.9.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the user data was written normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or is not a multiple of 8.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Write is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.
Abnormal termination [D]	FLMD error	18H (ST2)	A write error has occurred.
	Write error	1CH (ST2)	A write error has occurred.
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

7.9.4 Flowchart



7.9.5 Sample program

The following shows a sample program for Programming command processing.

```

/*****
/*
/* Write command (CSI-HS)
/*
/*****
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****
u16      fl_hs_write(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;
    u16    block_num;

    /*****
    /*      set params
    /*****
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    block_num = get_block_num(top, bottom); // get block num

    /*****
    /*      send command & check status
    /*****
    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;          // t.o. detected

    if (rc = put_cmd_hs(FL_COM_WRITE, 7, fl_cmd_prm)) // send "Programming" command
        return rc;                          // t.o. detected

    if (hs_busy_to(tWT3_MAX))
        return FLC_HSTO_ERR;          // t.o. detected

    rc = fl_hs_getstatus();             // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                        return rc; break; // case [B]
    }

    /*****
    /*      send user data
    /*****
    send_head = top;

    while(1){
        // make send data frame
        if ((bottom - send_head) > 256){ // rest size > 256 ?
            is_end = false;           // yes, not end frame
            send_size = 256;          // transmit size = 256 byte

```



```

    }
    else{
        is_end = true;
        send_size = bottom - send_head + 1;
                // transmit size = (bottom - send_head)+1 byte
    }
    memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                // set data frame payload
    send_head += send_size;

    if (hs_busy_to(tFD3_MAX)) // t.o. check before sending data frame
        return FLC_HSTO_ERR; // t.o. detected

    if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
                // send user data
        return rc;                // error detected

    if (hs_busy_to(tWT4_MAX))
        return FLC_HSTO_ERR;        // t.o. detected

    rc = fl_hs_getstatus();        // get status frame
    switch(rc) {
        case FLC_NO_ERR:            break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                    return rc; break; // case [B]
    }
    if (fl_st2 != FLST_ACK){        // ST2 = ACK ?
        rc = decode_status(fl_st2); // No
        return rc;                // case [D]
    }
    if (is_end)                    // send all user data ?
        break;                    // yes
}

/*****
/*      Check internally verify      */
/*****/
if (hs_busy_to(tWT5_MAX * block_num))
    return FLC_HSTO_ERR;        // t.o. detected

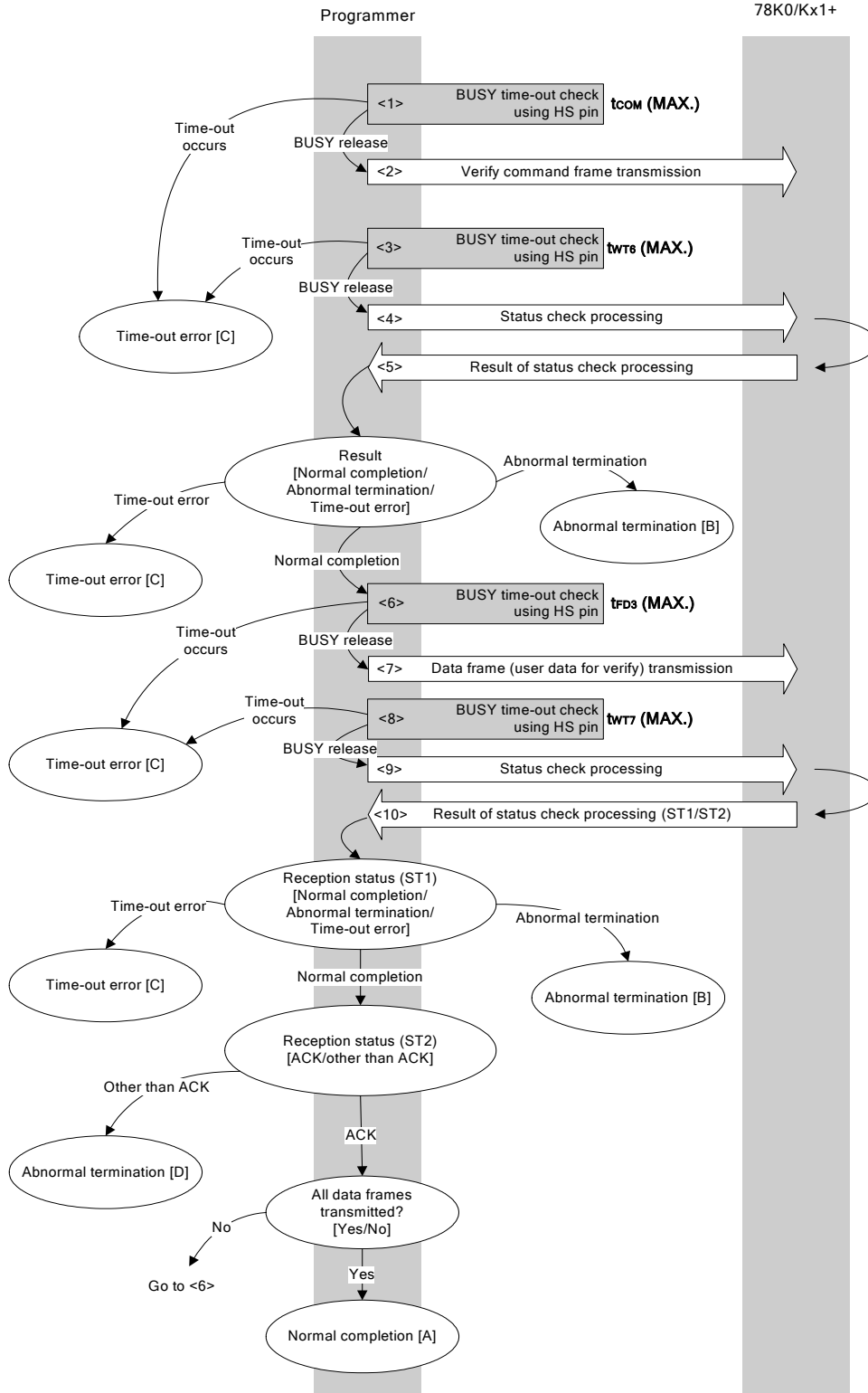
rc = fl_hs_getstatus();        // get status frame
// switch(rc) {
//     case FLC_NO_ERR: return rc; break; // case [A]
//     case FLC_HSTO_ERR: return rc; break; // case [C]
//     default: return rc; break; // case [B]
// }
return rc;
}

```

7.10 Verify Command

7.10.1 Processing sequence chart

Verify command processing sequence



7.10.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM}(MAX.)$).
- <2> The Verify command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT6}(MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

- <6> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD3}(MAX.)$).
- <7> User data for verifying is transmitted by data frame transmission processing.
- <8> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT7}(MAX.)$).
- <9> The status frame is acquired by status check processing.
- <10> The following processing is performed according to the result of status check processing (status code (ST1/ST2)) (also refer to the processing sequence chart and flowchart).

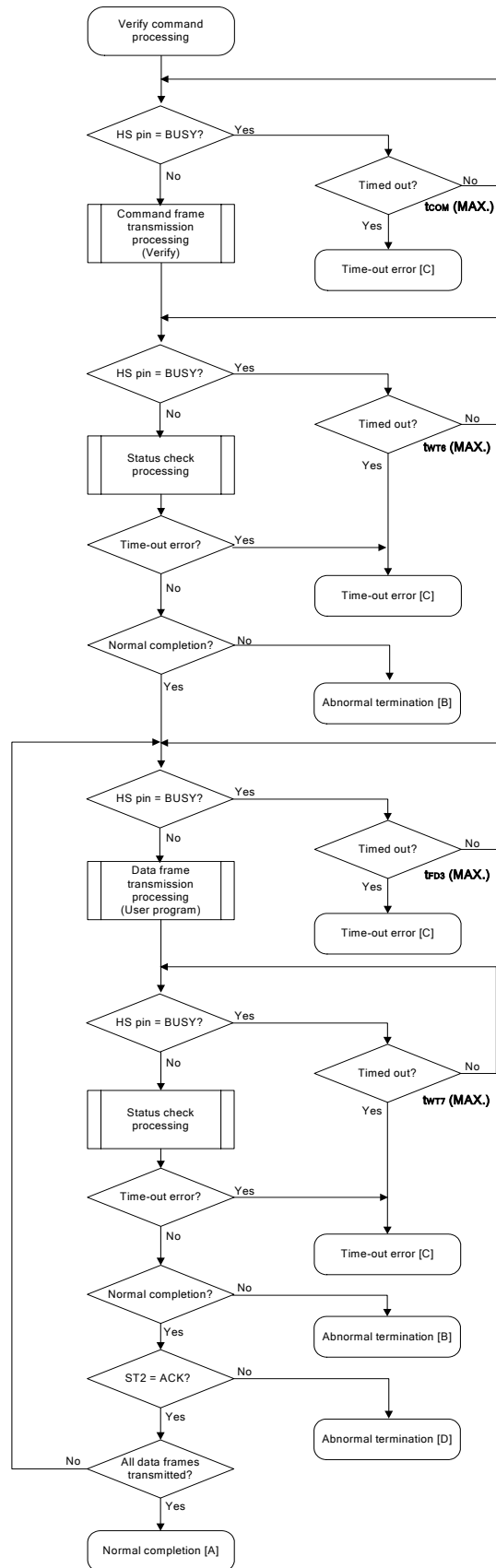
When ST1 = abnormal termination: Abnormal termination [B]
 When ST1 = time-out error: A time-out error [C] is returned.
 When ST1 = normal completion: The following processing is performed according to the ST2 value.

- When ST2 = ACK: If transmission of all data frames is completed, the processing ends normally [A].
If there still remain data frames to be transmitted, the processing re-executes the sequence from <6>.
- When ST2 ≠ ACK: Abnormal termination [D]

7.10.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the verify was completed normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame or data frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.
Abnormal termination [D]	Verify error	0FH (ST2)	The verify has failed, or another error has occurred.

7.10.4 Flowchart



7.10.5 Sample program

The following shows a sample program for Verify command processing.

```

/*****/
/*                                     */
/* Verify command (CSI-HS)           */
/*                                     */
/*****/
/* [i] u32 top      ... start address */
/* [i] u32 bottom  ... end address   */
/* [r] u16         ... error code     */
/*****/
u16      fl_hs_verify(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;

    /*****/
    /*      set params                    */
    /*****/
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    /*****/
    /*      send command & check status   */
    /*****/

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;           // t.o. detected

    if (rc = put_cmd_hs(FL_COM_VERIFY, 7, fl_cmd_prm)) // send "Verify" command
        return rc;                       // error detected

    if (hs_busy_to(tWT6_MAX))
        return FLC_HSTO_ERR;           // t.o. detected

    rc = fl_hs_getstatus();              // get status frame
    switch(rc) {
        case  FLC_NO_ERR:                 break; // continue
    // case  FLC_HSTO_ERR: return rc;     break; // case [C]
        default:                          return rc;   break; // case [B]
    }

    /*****/
    /*      send user data                */
    /*****/

```

```

send_head = top;

while(1){

    // make send data frame
    if ((bottom - send_head) > 256){ // rest size > 256 ?
        is_end = false;           // yes, not is_end frame
        send_size = 256;         // transmit size = 256 byte
    }
    else{
        is_end = true;
        send_size = bottom - send_head + 1;
                               // transmit size = (bottom - send_head)+1 byte
    }

    memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                               // set data frame payload
    send_head += send_size;

    if (hs_busy_to(tFD3_MAX))
        return FLC_HSTO_ERR;    // t.o. detected

    if (rc = put_dfrm_hs(send_size, fl_txdata_frm, is_end))
                               // send user data
        return rc;             // error detected

    if (hs_busy_to(tWT7_MAX))
        return FLC_HSTO_ERR;    // t.o. detected

    rc = fl_hs_getstatus();     // get status frame
    switch(rc) {
        case FLC_NO_ERR:        break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                return rc; break; // case [B]
    }

    if (fl_st2 != FLST_ACK){    // ST2 = ACK ?
        rc = decode_status(fl_st2); // No
        return rc;              // case [D]
    }

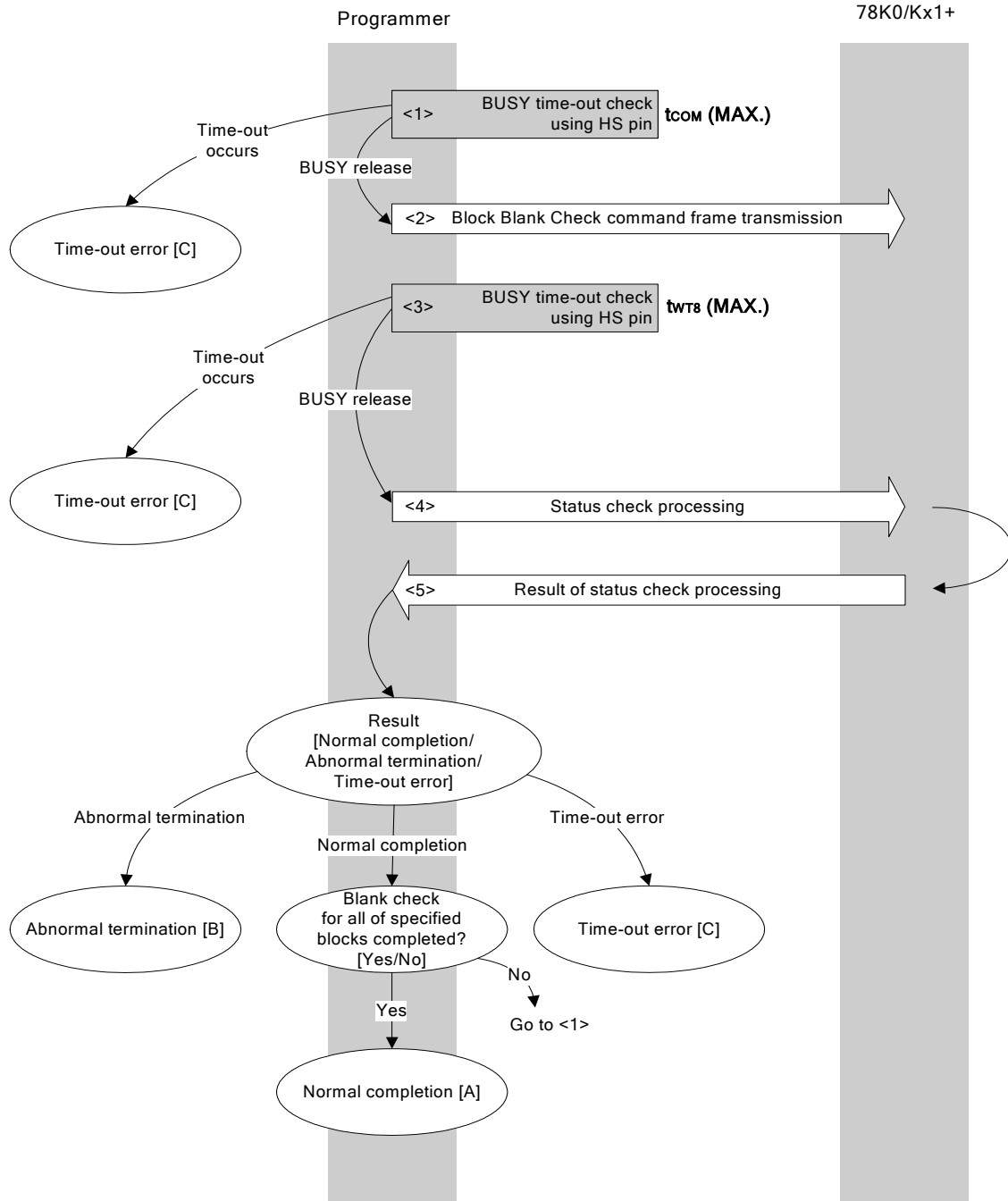
    if (is_end)                 // send all user data ?
        break;                  // yes
}
return FLC_NO_ERR; // case [A]
}

```

7.11 Block Blank Check Command

7.11.1 Processing sequence chart

Block Blank Check command processing sequence



7.11.2 Description of processing sequence

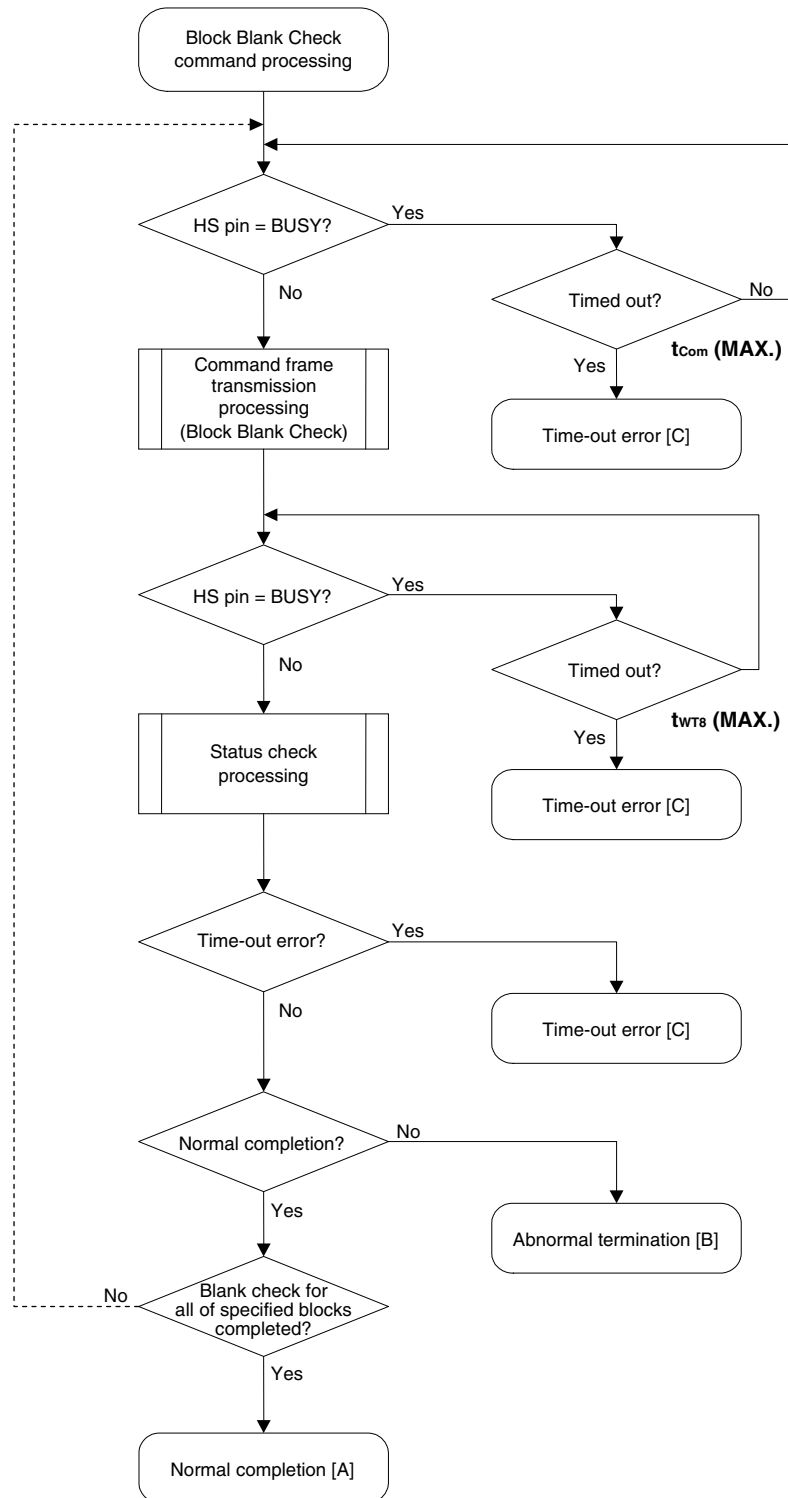
- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM}(MAX.)$).
- <2> The Block Blank Check command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT8}(MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When a time-out error occurs: A time-out error [C] is returned.
 When the processing ends abnormally: Abnormal termination [B]
 When the processing ends normally: If the blank check for all of the specified blocks is not yet completed, processing changes the block number and re-executes the sequence from <1>.
 If the blank check for all of the specified blocks is completed, the processing ends normally [A].

7.11.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and all of the specified blocks are blank.
Abnormal termination [B]	Parameter error	05H	The block number is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	MRG11 error	1BH	The specified block in the flash memory is not blank.
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.

7.11.4 Flowchart



7.11.5 Sample program

The following shows a sample program for Block Blank Check command processing for one block.

```

/*****
/*
/* Block blank check command (CSI-HS)
/*
/*****
/* [i] u16 block ... block number
/* [r] u16 ... error code
/*****
u16 fl_hs_blk_blank_chk(u8 block)
{
    u16 rc;

    fl_cmd_prm[0] = block; // "BLK"

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm))
        // send "Block Blank Check" command
        return rc; // case [C]

    if (hs_busy_to(tWT8_MAX))
        return FLC_HSTO_ERR; // t.o. detected :case [C]

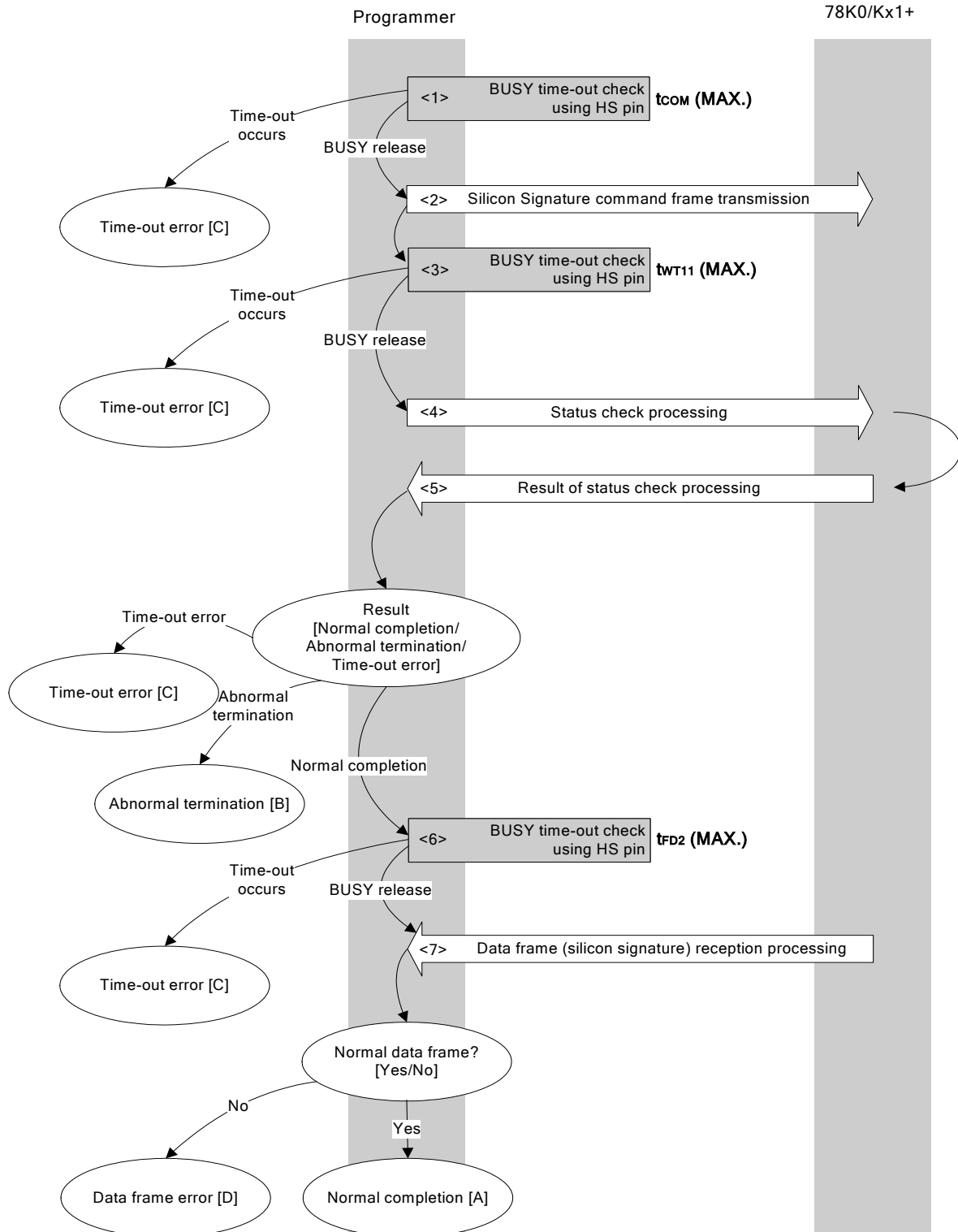
    rc = fl_hs_getstatus(); // get status frame
    // switch(rc) {
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_HSTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}

```

7.12 Silicon Signature Command

7.12.1 Processing sequence chart

Silicon Signature command processing sequence



7.12.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM(MAX)}$).
- <2> The Silicon Signature command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT11(MAX)}$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

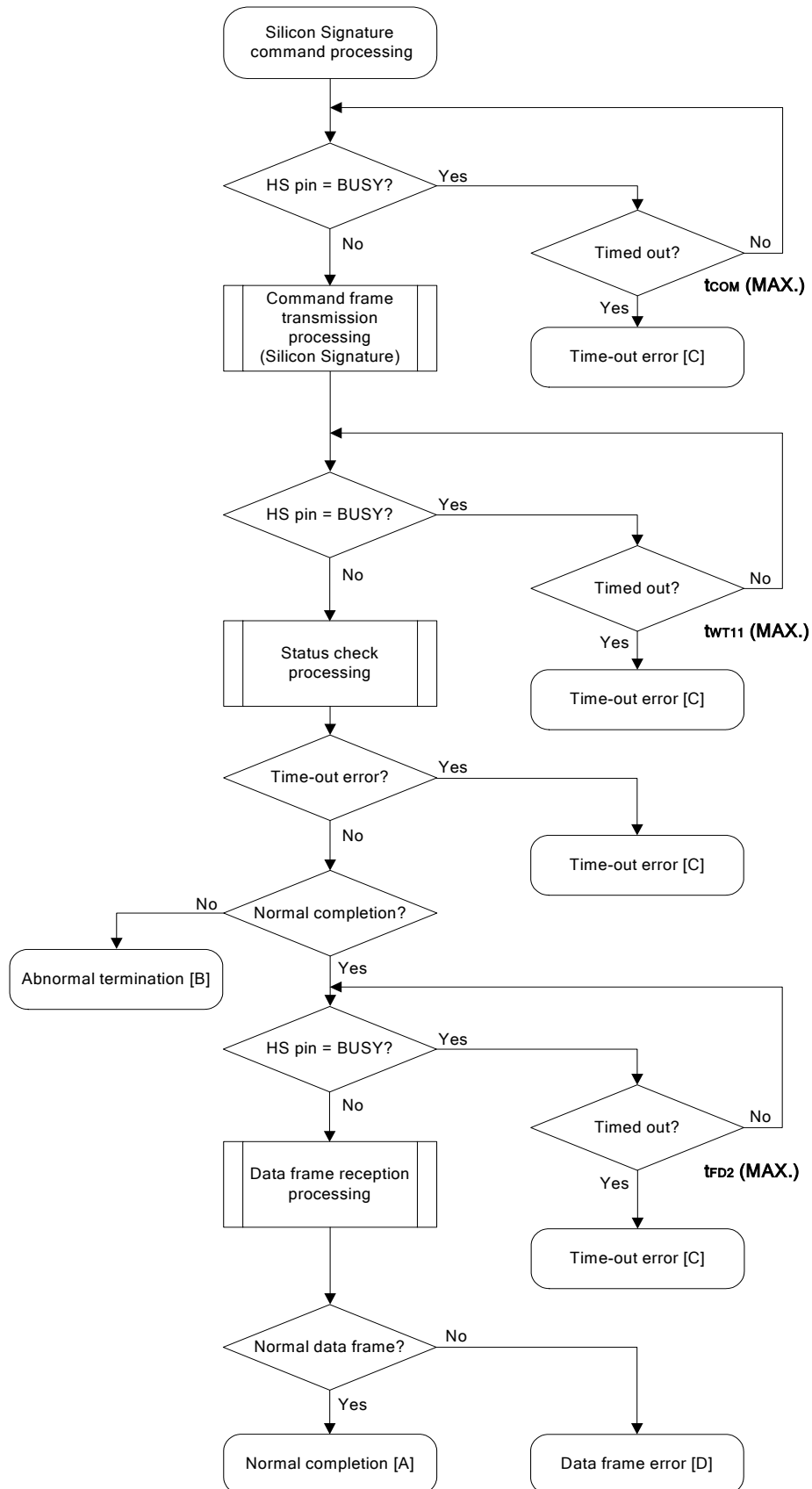
- <6> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD2(MAX)}$).
- <7> The received data frame (silicon signature data) is checked.

If data frame is normal: Normal completion [A]
 If data frame is abnormal: Data frame error [D]

7.12.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the silicon signature was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.
Data frame error [D]		–	The checksum of the data frame received as silicon signature data does not match.

7.12.4 Flowchart



7.12.5 Sample program

The following shows a sample program for Silicon Signature command processing.

```

/*****
/*
/* Get silicon signature command (CSI-HS)
/*
/*****
/* [i] u8 *sig... pointer to signature save area
/* [r] u16 ... error code
/*****
u16      fl_hs_getsig(u8 *sig)
{
    u16    rc;

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm))
        return rc;                // send "Silicon Signature" command
        // error detected :case [C]

    if (hs_busy_to(tWT11_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    rc = fl_hs_getstatus();        // get status frame
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                  return rc; break; // case [B]
    }

    if (hs_busy_to(tFD2_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    rc = get_dfrm_hs(fl_rxdata_frm); // get signature data

    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                  return rc; break; // case [D]
    }

    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
        // copy Signature data

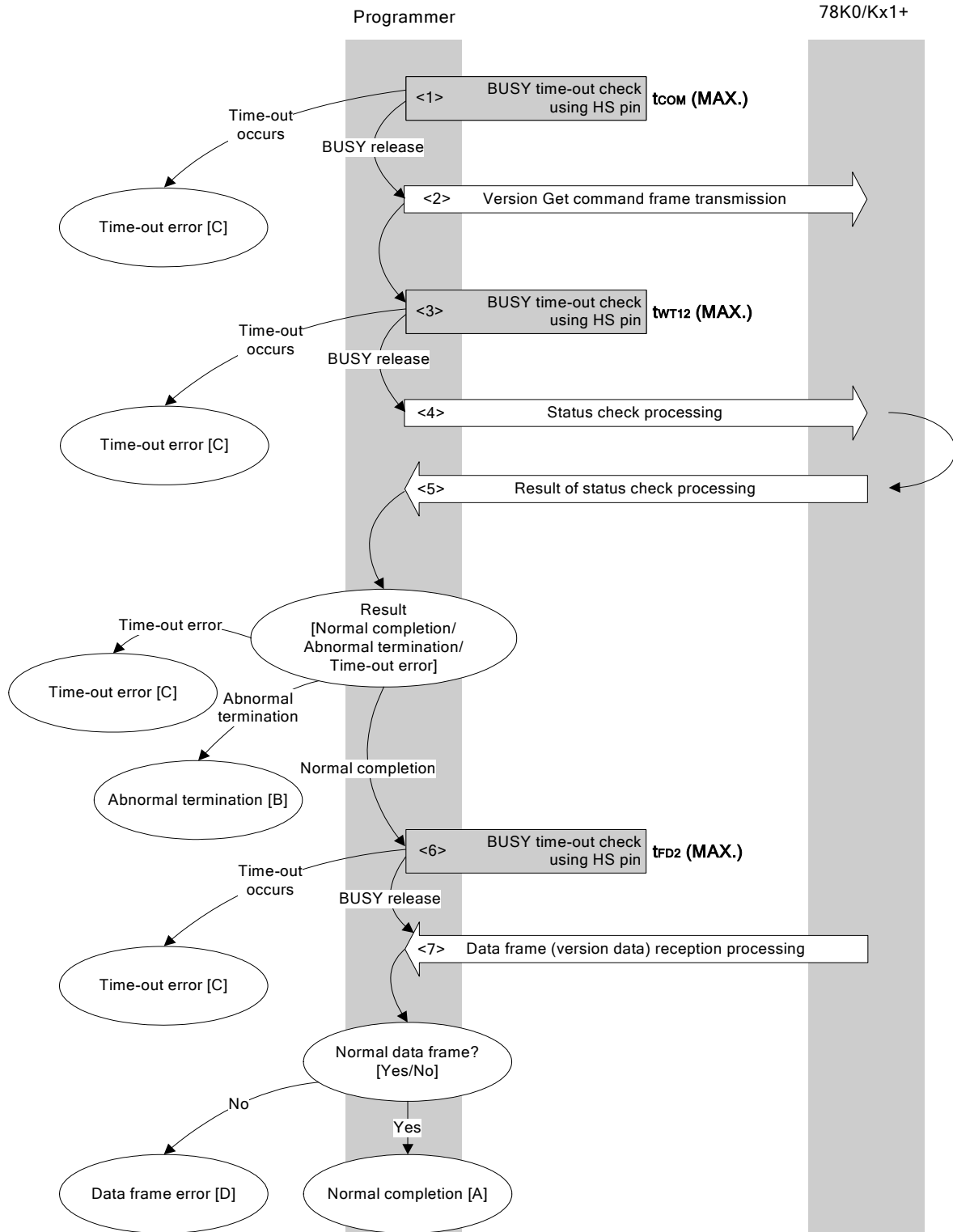
    return rc;                    // case [A]
}

```

7.13 Version Get Command

7.13.1 Processing sequence chart

Version Get command processing sequence



7.13.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM(MAX)}$).
- <2> The Version Get command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT12(MAX)}$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

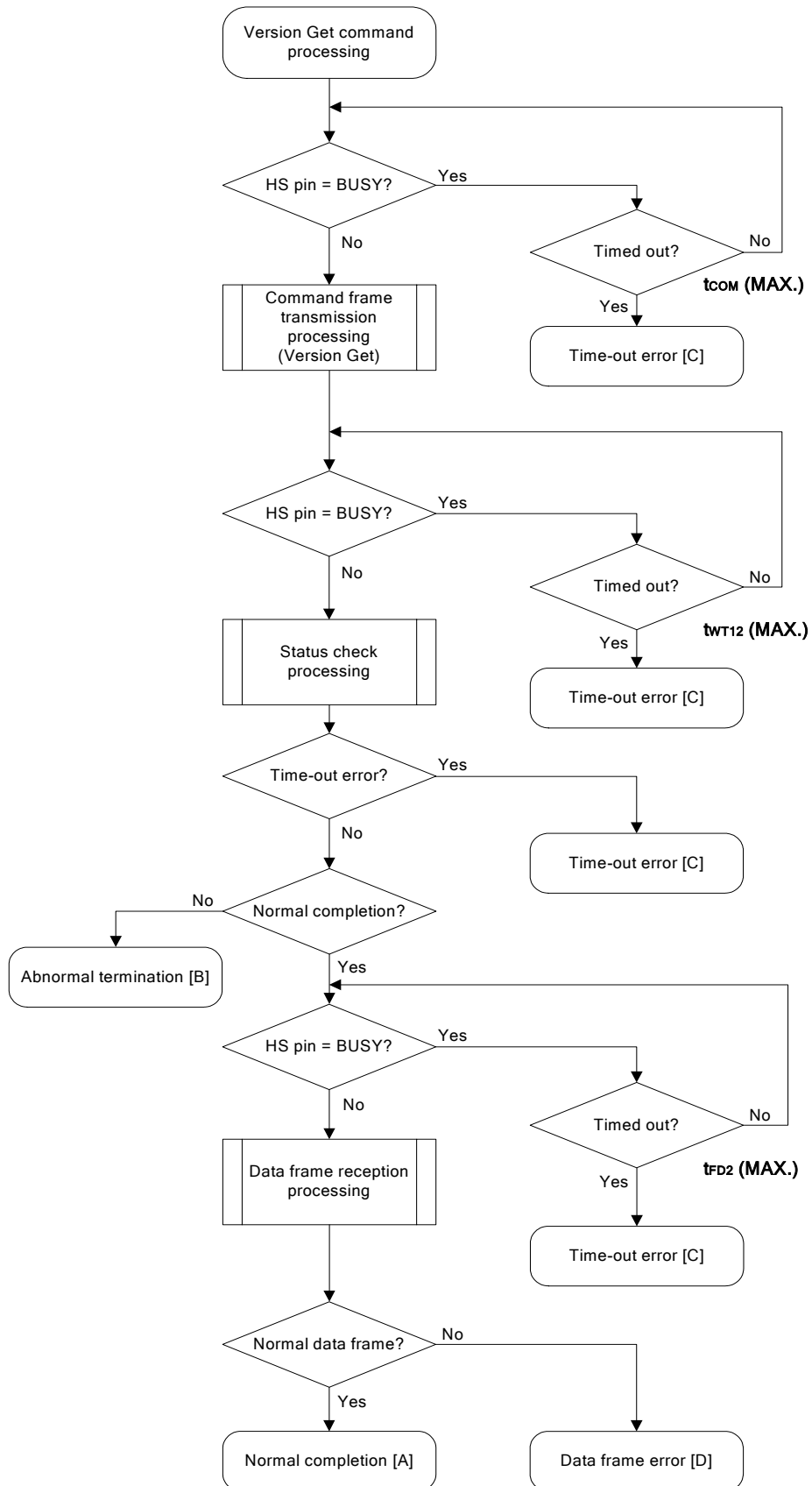
- <6> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD2(MAX)}$).
- <7> The received data frame (version data) is checked.

If data frame is normal: Normal completion [A]
 If data frame is abnormal: Data frame error [D]

7.13.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and version data was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.
Data frame error [D]		–	The checksum of the data frame received as version data does not match.

7.13.4 Flowchart



7.13.5 Sample program

The following shows a sample program for Version Get command processing.

```

/*****
/*
/* Get device/firmware version command (CSI-HS)
/*
/*
/*****
/* [i] u8 *buf      ... pointer to version data save area
/* [r] u16          ... error code
/*****
u16      fl_hs_getver(u8 *buf)
{
    u16    rc;

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_GET_VERSION, 1, fl_cmd_prm))
        // send "Version Get" command
        return rc;                // error detected :case [C]

    if (hs_busy_to(tWT12_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    rc = fl_hs_getstatus();        // get status frame
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                  return rc; break; // case [B]
    }

    if (hs_busy_to(tFD2_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    rc = get_dfrm_hs(fl_rxddata_frm); // get signature data
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                  return rc; break; // case [D]
    }

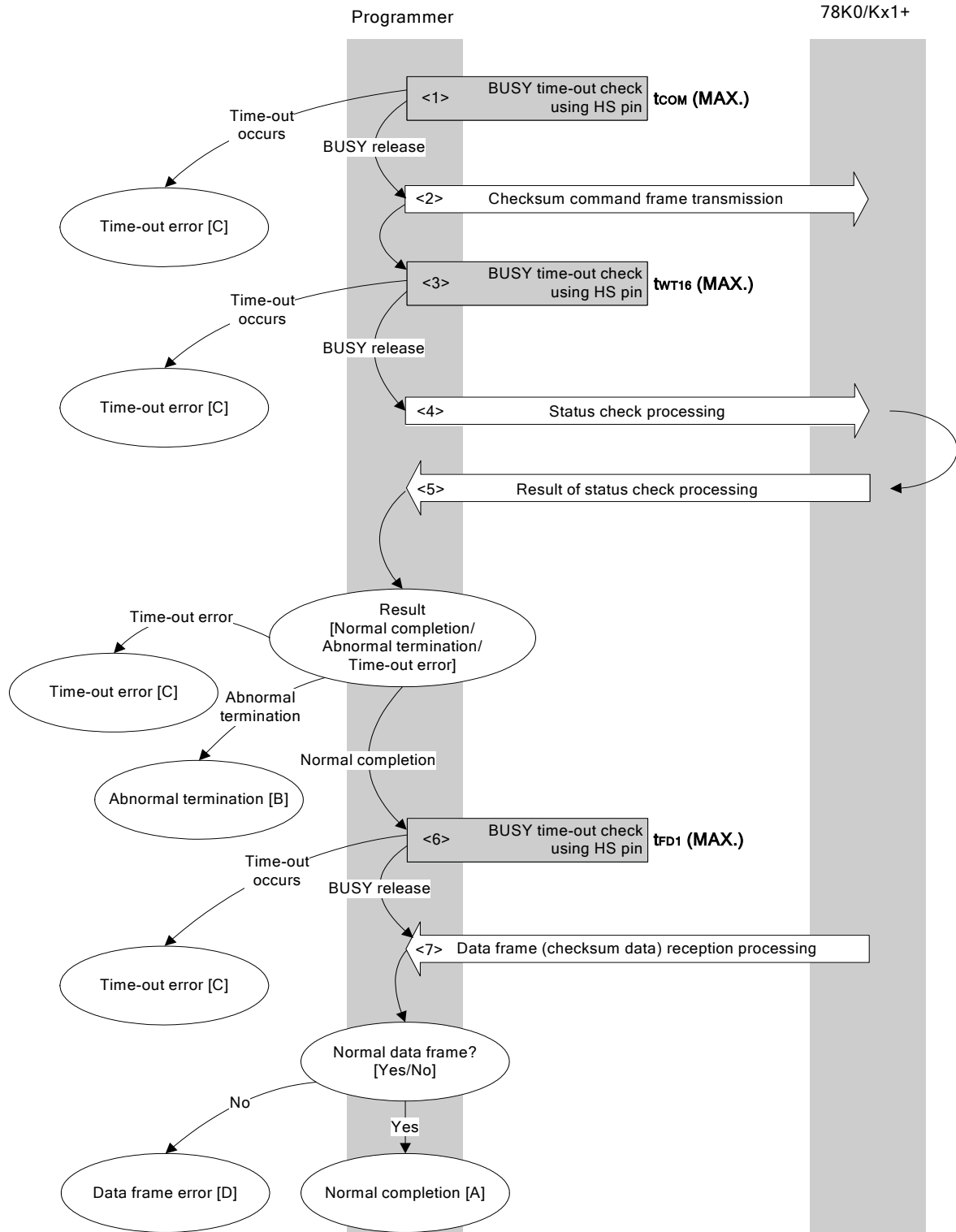
    memcpy(buf, fl_rxddata_frm+OFS_STA_PLD, DFV_LEN); // copy version data
    return rc;                    // case [A]
}

```

7.14 Checksum Command

7.14.1 Processing sequence chart

Checksum command processing sequence



7.14.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM(MAX)}$).
- <2> The Checksum command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT16(MAX)}$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

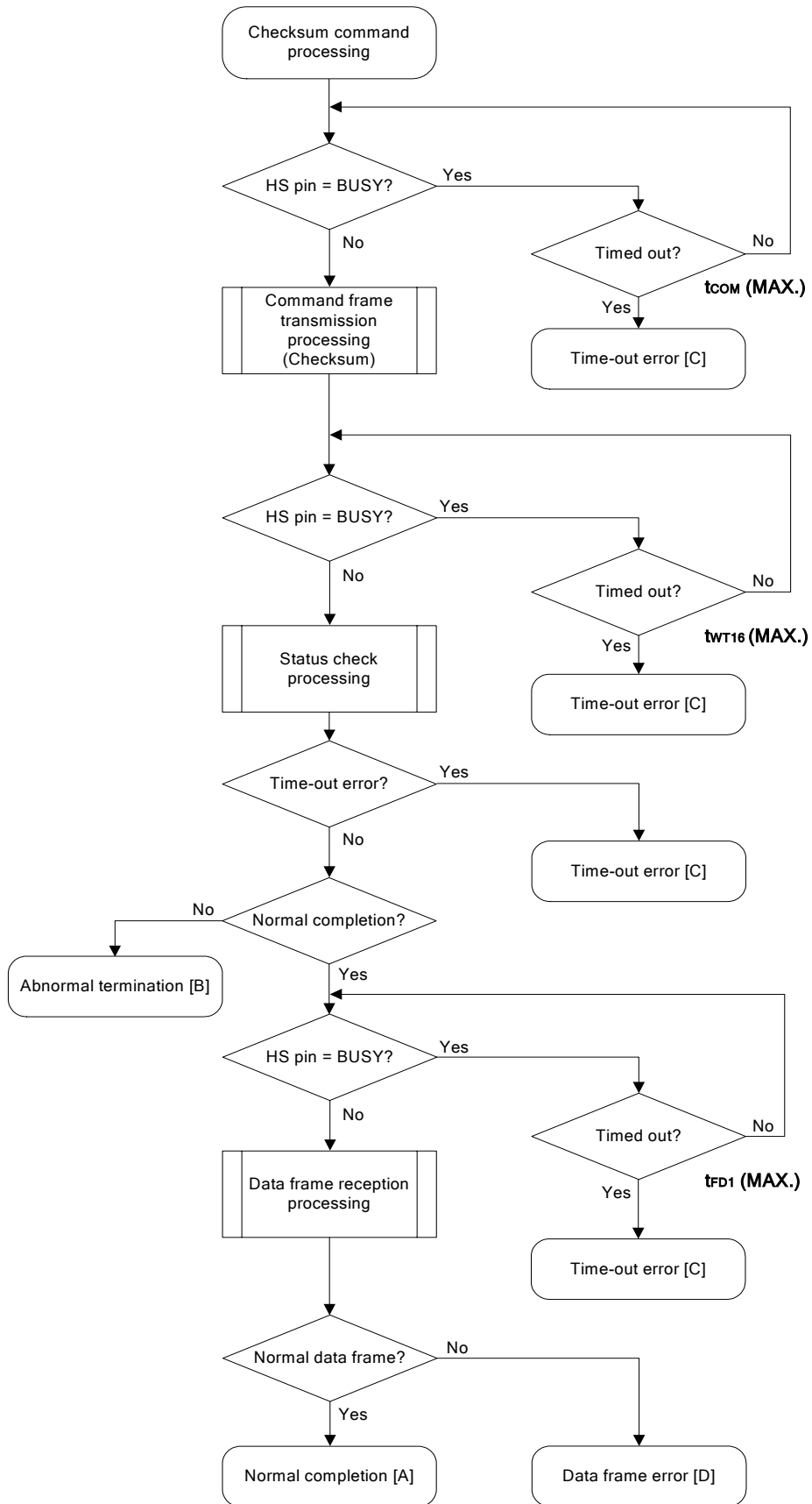
- <6> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD1(MAX)}$).
- <7> The received data frame (checksum data) is checked.

If data frame is normal: Normal completion [A]
 If data frame is abnormal: Data frame error [D]

7.14.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and checksum data was acquired normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.
Data frame error [D]		–	The checksum of the data frame received as version data does not match.

7.14.4 Flowchart



7.14.5 Sample program

The following shows a sample program for Checksum command processing.

```

/*****
/*
/* Get checksum command (CSI-HS)
/*
/*****
/* [i] u16 *sum    ... pointer to checksum save area
/* [i] u32 top    ... start address
/* [i] u32 bottom ... end address
/* [r] u16        ... error code
/*****
u16    fl_hs_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16    rc;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;           // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm))
        return rc;                     // send "Checksum" command
                                        // error detected :case [C]

    if (hs_busy_to(tWT16_MAX))
        return FLC_HSTO_ERR;           // t.o. detected :case [C]

    rc = fl_hs_getstatus();             // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                        return rc; break; // case [B]
    }

    if (hs_busy_to(tFD1_MAX))
        return FLC_HSTO_ERR;           // t.o. detected :case [C]

    rc = get_dfrm_hs(fl_rxdata_frm);    // get signature data

    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                        return rc; break; // case [D]
    }

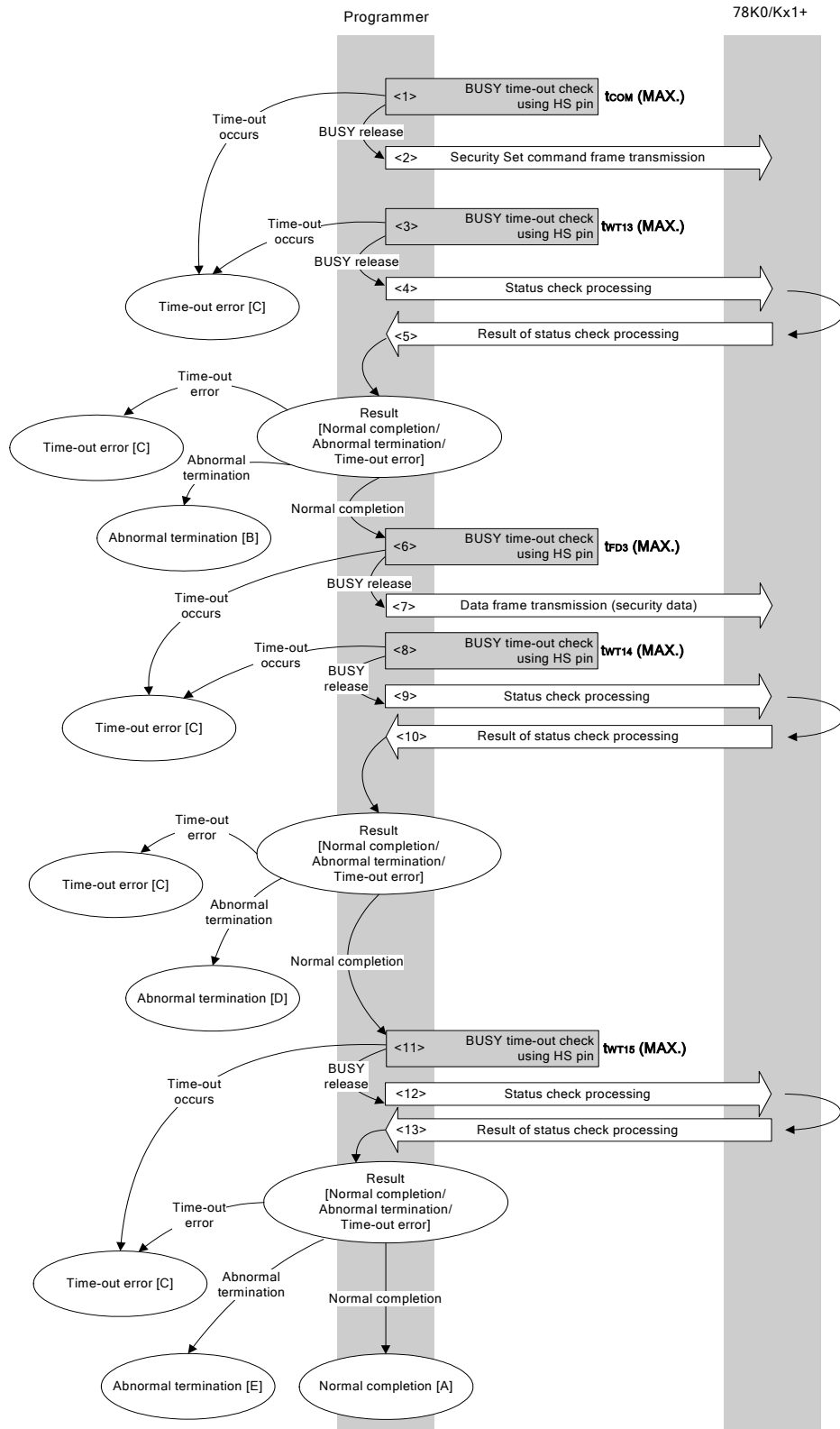
    *sum = (fl_rxdata_frm[OFS_STA_PLD] << 8) + fl_rxdata_frm[OFS_STA_PLD+1];
                                        // set SUM data
    return rc;                          // case [A]
}

```

7.15 Security Set Command

7.15.1 Processing sequence chart

Security Set command processing sequence



7.15.2 Description of processing sequence

- <1> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{COM}(MAX.)$).
- <2> The Security Set command is transmitted by command frame transmission processing.
- <3> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT13}(MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

- <6> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{FD3}(MAX.)$).
- <7> The data frame (security setting data) is transmitted by data frame transmission processing.
- <8> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT14}(MAX.)$).
- <9> The status frame is acquired by status check processing.
- <10> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <11>.
 When the processing ends abnormally: Abnormal termination [D]
 When a time-out error occurs: A time-out error [C] is returned.

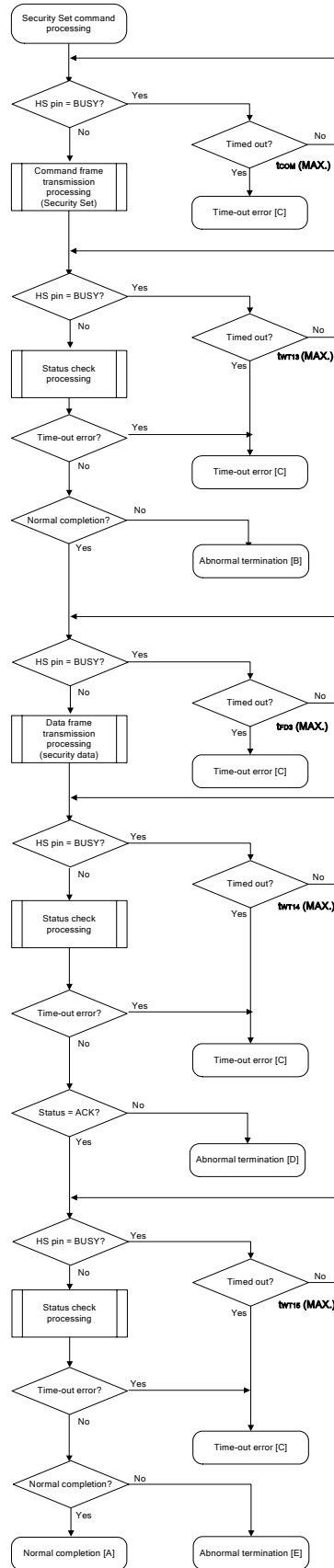
- <11> A 78K0/Kx1+ BUSY status is checked using the HS pin.
If a BUSY time-out occurs, a time-out error [C] is returned (time-out time $t_{WT15}(MAX.)$).
- <12> The status frame is acquired by status check processing.
- <13> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
 When the processing ends abnormally: Abnormal termination [E]
 When a time-out error occurs: A time-out error [C] is returned.

7.15.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and security setting was performed normally.
Abnormal termination [B]	Parameter error	05H	BLK or PAG is not 00H.
	Checksum error	07H	The checksum of the transmitted command frame or data frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Processing timed out due to the busy status at the HS pin.
Abnormal termination [D]	FLMD error	18H	A write error has occurred.
	Write error	1CH	A write error has occurred (including the case of security data already being set).
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

7.15.4 Flowchart



7.15.5 Sample program

The following shows a sample program for Security Set command processing.

```

/*****
/*
/* Set security flag command (CSI-HS)
/*
/*****
/* [i] u8 scf      ... Security flag data
/* [r] u16         ... error code
/*****
u16      fl_hs_setscf(u8 scf)
{
    u16    rc;

    fl_cmd_prm[0] = 0x00;          // "BLK" (must be 0x00)
    fl_cmd_prm[1] = 0x00;          // "PAG" (must be 0x00)
    fl_txdata_frm[0] = scf | 0b11111000;
                                   // "FLG" (upper 5bits must be '1' (to make sure))

    if (hs_busy_to(tCOM_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    if (rc = put_cmd_hs(FL_COM_SET_SECURITY, 3, fl_cmd_prm))
                                   // send "Security Set" command
        return rc;                // error detected :case [C]

    if (hs_busy_to(tWT13_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    rc = fl_hs_getstatus();        // get status frame
    switch(rc) {
        case FLC_NO_ERR:           break; // continue
        // case FLC_HSTO_ERR: return rc; break; // case [C]
        default:                   return rc; break; // case [B]
    }

    if (hs_busy_to(tFD3_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

    if (rc = put_dfrm_hs(1, fl_txdata_frm, true)) // send securithi setting data
        return rc;                // error detected :case [C]

    if (hs_busy_to(tWT14_MAX))
        return FLC_HSTO_ERR;      // t.o. detected :case [C]

```

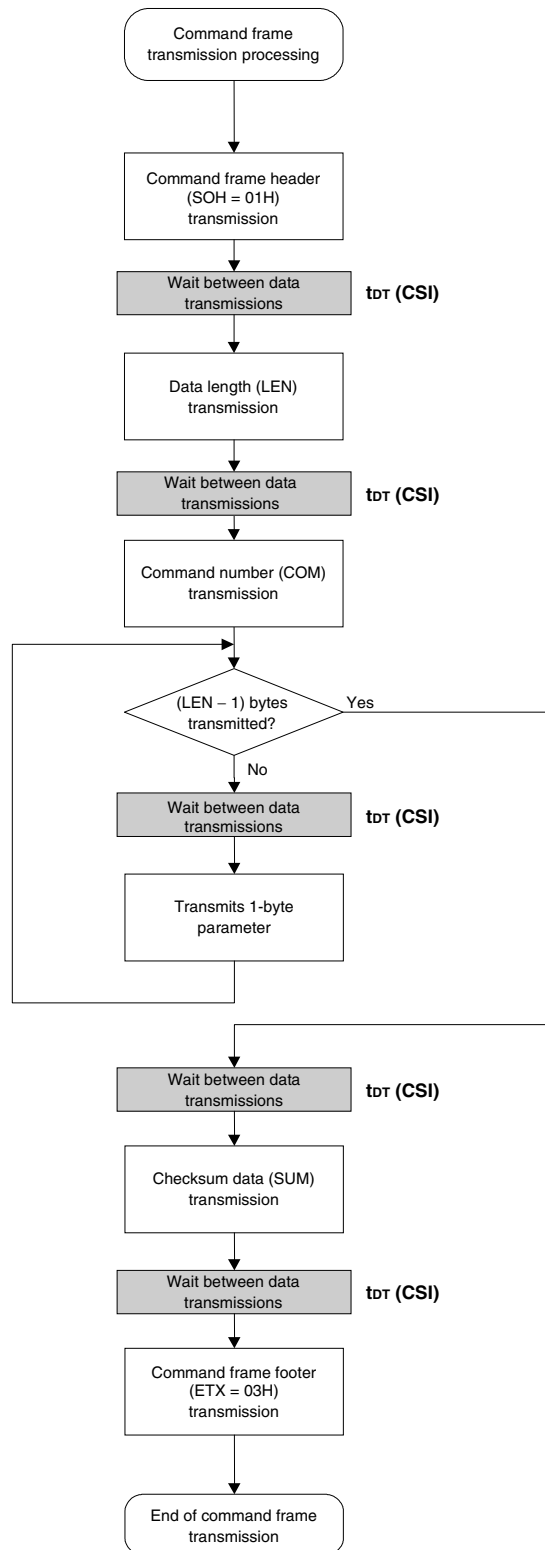
```
rc = fl_hs_getstatus();           // get status frame
switch(rc) {
    case FLC_NO_ERR:              break; // continue
//   case FLC_HSTO_ERR: return rc; break; // case [C]
    default:                      return rc; break; // case [B]
}

if (hs_busy_to(tWT15_MAX))
    return FLC_HSTO_ERR;          // t.o. detected

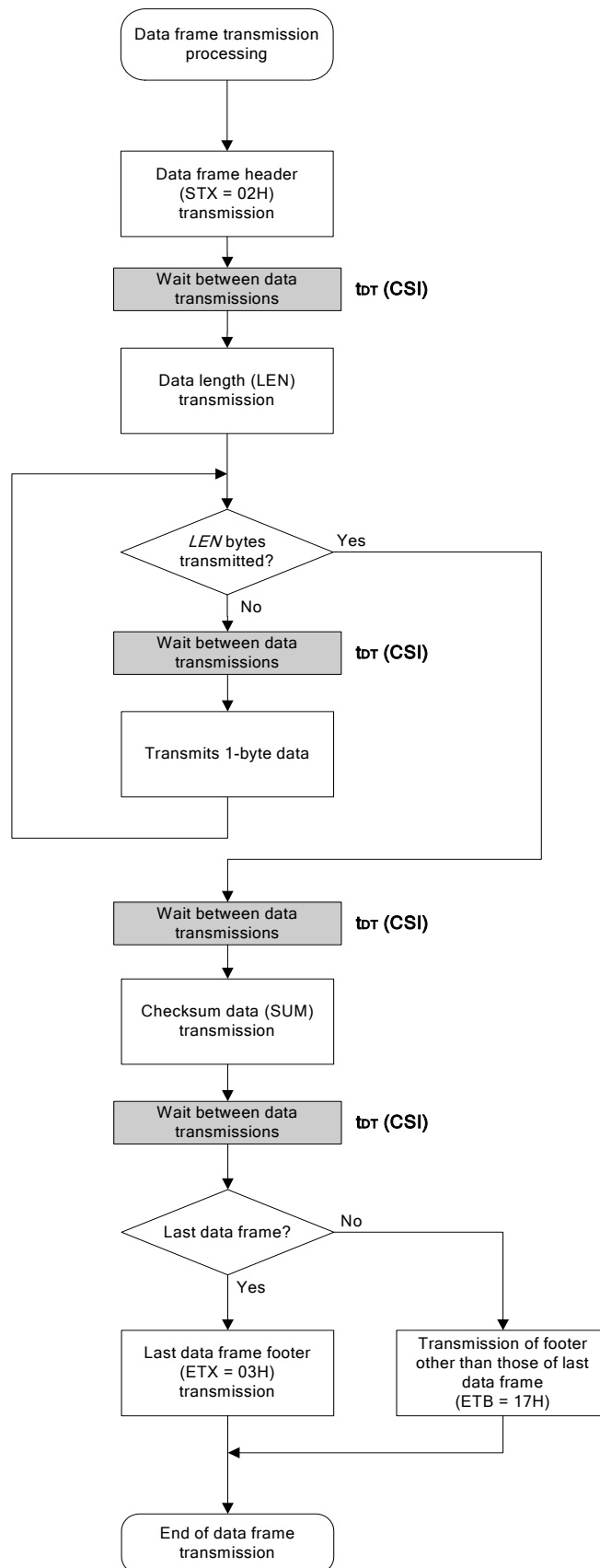
rc = fl_hs_getstatus();           // get status frame again
// switch(rc) {
//   case FLC_NO_ERR:  return rc;  break; // case [A]
//   case FLC_HSTO_ERR: return rc;  break; // case [C]
//   default:         return rc;   break; // case [B]
// }
return rc;
}
```

CHAPTER 8 3-WIRE SERIAL I/O COMMUNICATION MODE (CSI)

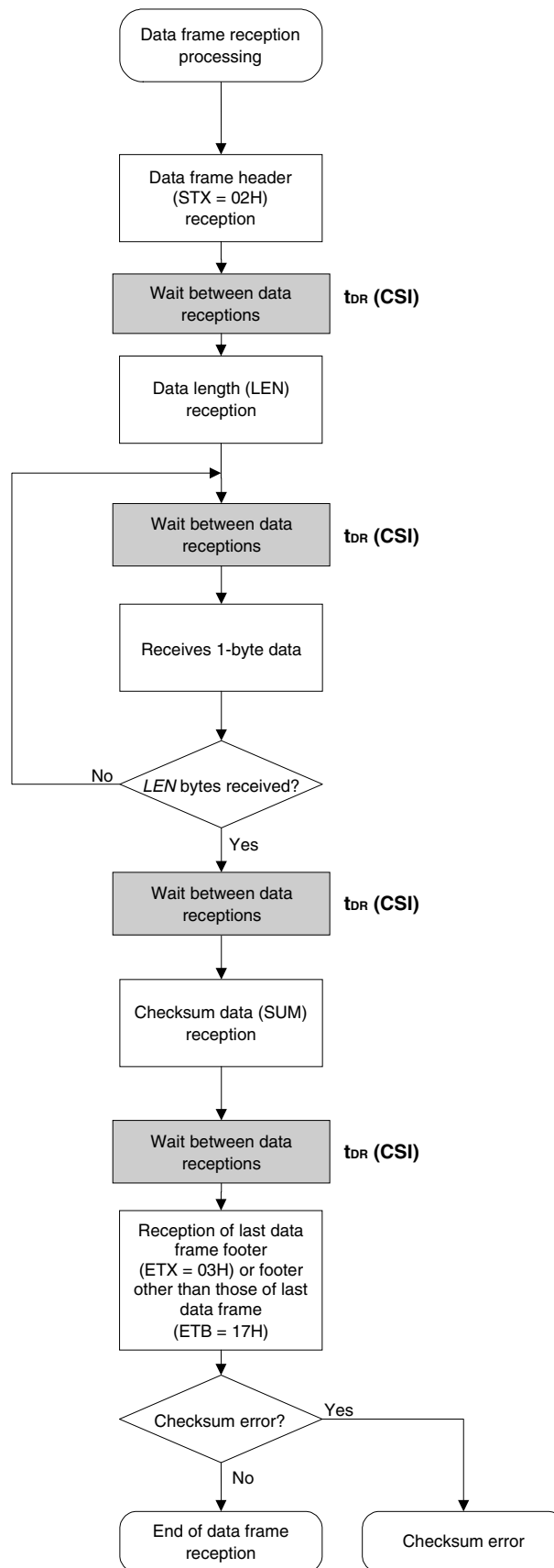
8.1 Command Frame Transmission Processing Flowchart



8.2 Data Frame Transmission Processing Flowchart



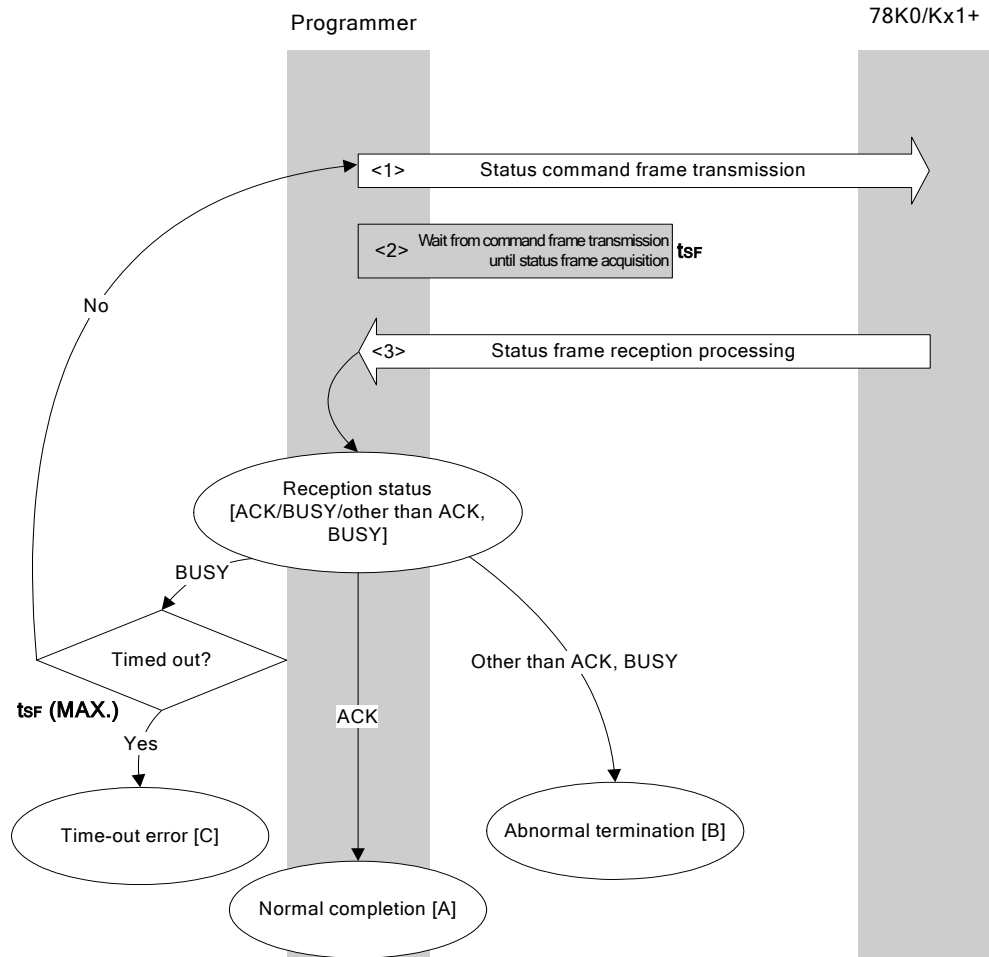
8.3 Data Frame Reception Processing Flowchart



8.4 Status Command

8.4.1 Processing sequence chart

Status command processing sequence



8.4.2 Description of processing sequence

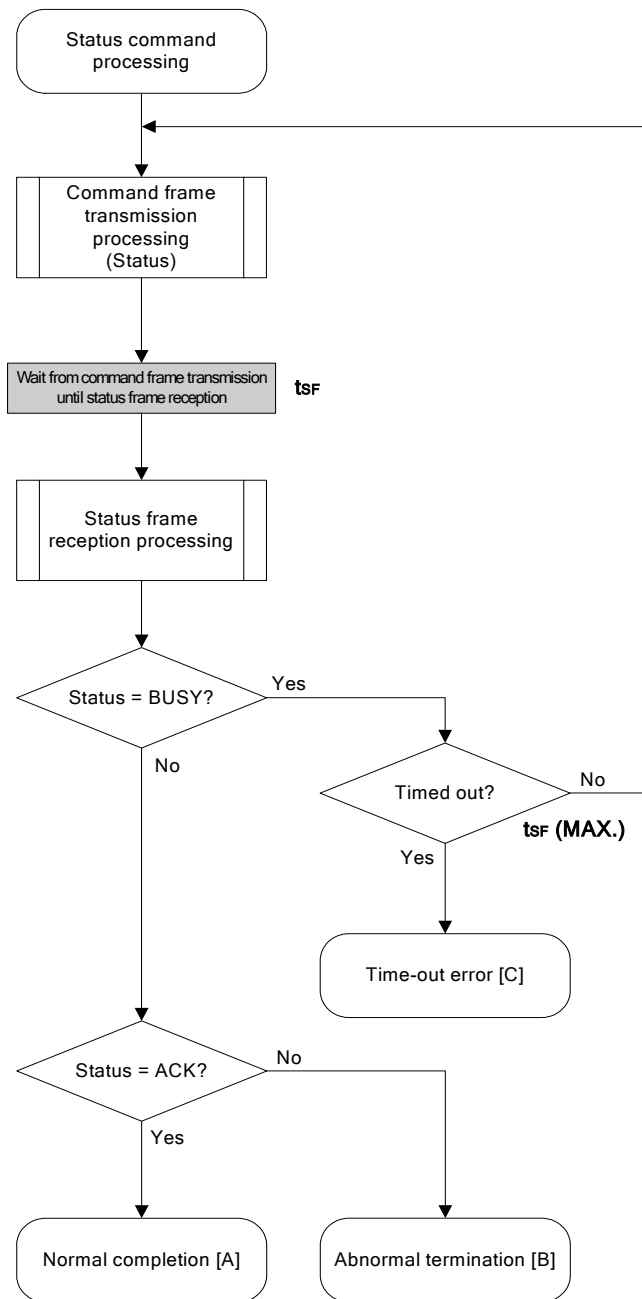
- <1> The Status command is transmitted by command frame transmission processing.
- <2> Waits from command transmission until status frame reception (wait time t_{SF}).
- <3> The status code is checked.

When ST1 = ACK: Normal completion [A]
 When ST1 = BUSY: A time-out check is performed ($t_{SF} (MAX.)$).
 If the processing is not timed out, the sequence is re-executed from <1>.
 If a time-out occurs, a time-out error [C] is returned.
 When ST1 ≠ ACK, BUSY: Abnormal termination [B]

8.4.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The status frame transmitted from the 78K0/Kx1+ has been received normally.
Abnormal termination [B]	Command error	04H	An unsupported command or abnormal frame has been received.
	Parameter error	05H	Command information (parameter) is invalid.
	Checksum error	07H	The data of the frame transmitted from the programmer is abnormal.
	Verify error	0FH	A verify error has occurred for the data of the frame transmitted from the programmer.
	Protect error	10H	An attempt was made to execute processing prohibited by the Security Set command.
	Negative acknowledgment (NACK)	15H	Negative acknowledgment
	FLMD error	18H	A write error has occurred.
	MRG10 error	1AH	An erase error has occurred.
	MRG11 error	1BH	An internal verify error has occurred during data write, or a blank check error has occurred.
	Write error	1CH	A write error has occurred.
Time-out error [C]		–	After command transmission, the specified time has elapsed but a BUSY response is still returned.

8.4.4 Flowchart



8.4.5 Sample program

The following shows a sample program for Status command processing.

```

/*****
/*
/* Get status command (CSI)
/*
/*****
/* [r] u16      ... decoded status or error code
/*
/* (see fl.h/fl-proto.h &
/*      definition of decode_status() in fl.c)
/*****
static u16 fl_csi_getstatus(void)
{
    u16    rc;

    start_flto(tSF_MAX);

    while(1){

        put_cmd_csi(FL_COM_GET_STA, 1, fl_cmd_prm);    // send "Status" command frame
        fl_wait(tSF);                                  // wait

        rc = get_sfrm_csi(fl_rxddata_frm);             // get status frame

        switch(rc){
            case FLC_BUSY:
                if (check_flto())                      // time out ?
                    return FLC_DFTO_ERR;              // Yes, time-out // case [C]
                continue;                              // No, retry

            default:
                return rc;                             // checksum error

            case FLC_NO_ERR:                            // no error
                break;

        }
        if (fl_st1 == FLST_BUSY){ // ST1 = BUSY
            if (check_flto())    // time out ?
                return FLC_DFTO_ERR; // Yes, time-out // case [C]
            continue;          // No, retry
        }
        break;                // ACK or other error (but BUSY)
    }

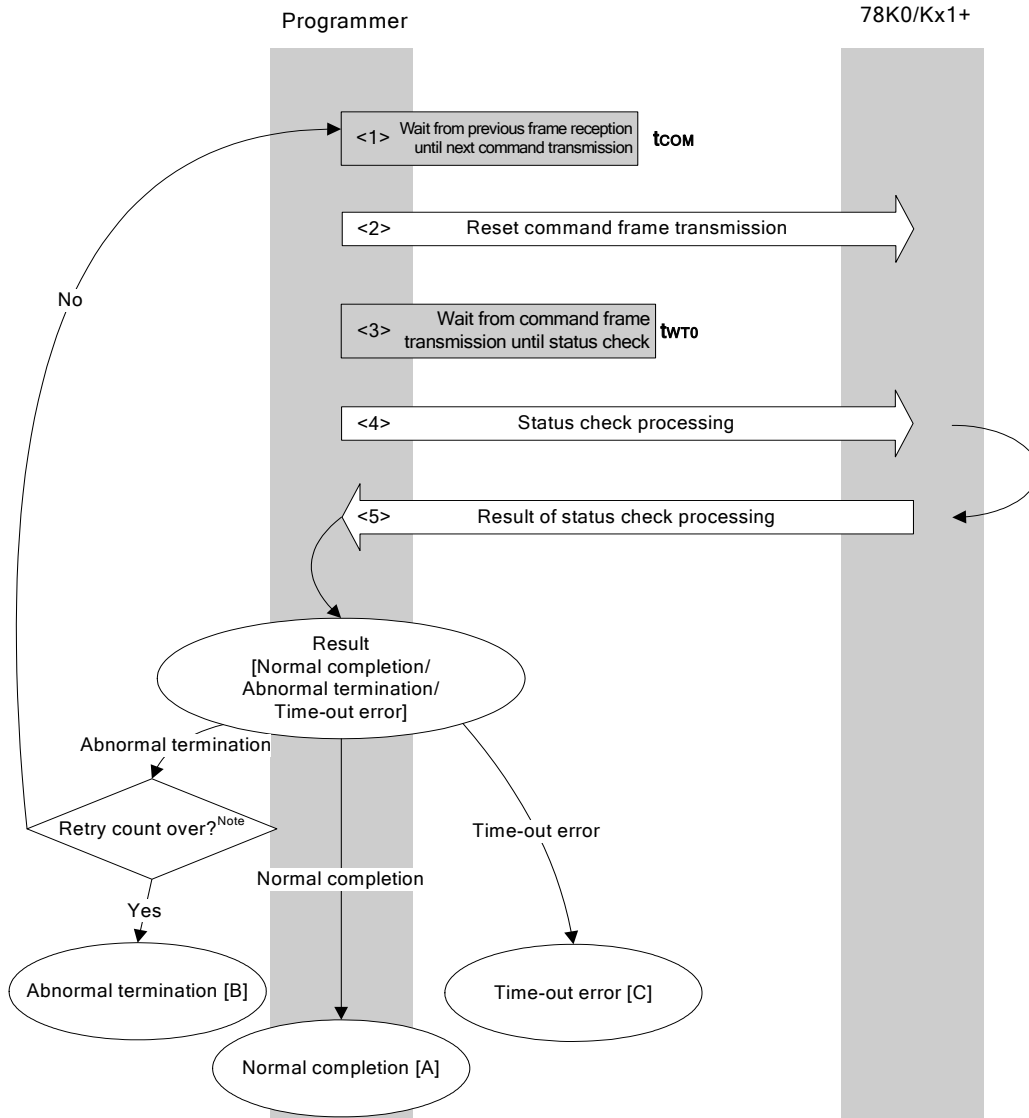
    rc = decode_status(fl_st1);    // decode status to return code
// switch(rc) {
//
//     case FLC_NO_ERR: return rc; break; // case [A]
//     default:        return rc; break; // case [B]
// }
    return rc;
}

```

8.5 Reset Command

8.5.1 Processing sequence chart

Reset command processing sequence



Note Do not exceed the retry count for the reset command transmission (up to 16 times).

8.5.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Reset command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WTO} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]

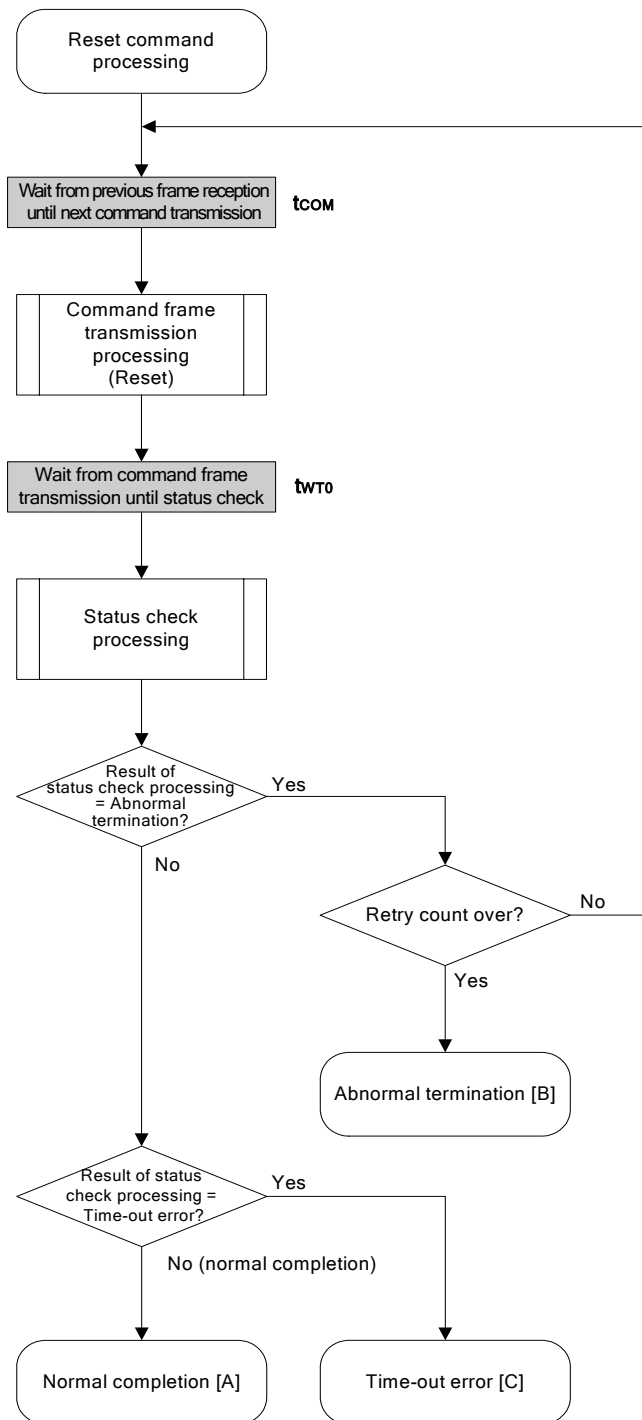
When the processing ends abnormally: The sequence is re-executed from <1> if the retry count is not over.
If the retry count is over, the processing ends abnormally [B].

When a time-out error occurs: A time-out error [C] is returned.

8.5.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and synchronization between the programmer and the 78K0/Kx1+ has been established.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	Status check processing timed out.

8.5.4 Flowchart



8.5.5 Sample program

The following shows a sample program for Reset command processing.

```

/*****
/*
/* Reset command (CSI)
/*
/*
/*****
/* [r] u16      ... error code
/*****
u16      fl_csi_reset(void)
{
    u16    rc;
    u32    retry;

    for (retry = 0; retry < tRS; retry++){

        fl_wait(tCOM);                // wait before sending command frame

        put_cmd_csi(FL_COM_RESET, 1, fl_cmd_prm);    // send "Reset" command frame

        fl_wait(tWT0);

        rc = fl_csi_getstatus();        // get status

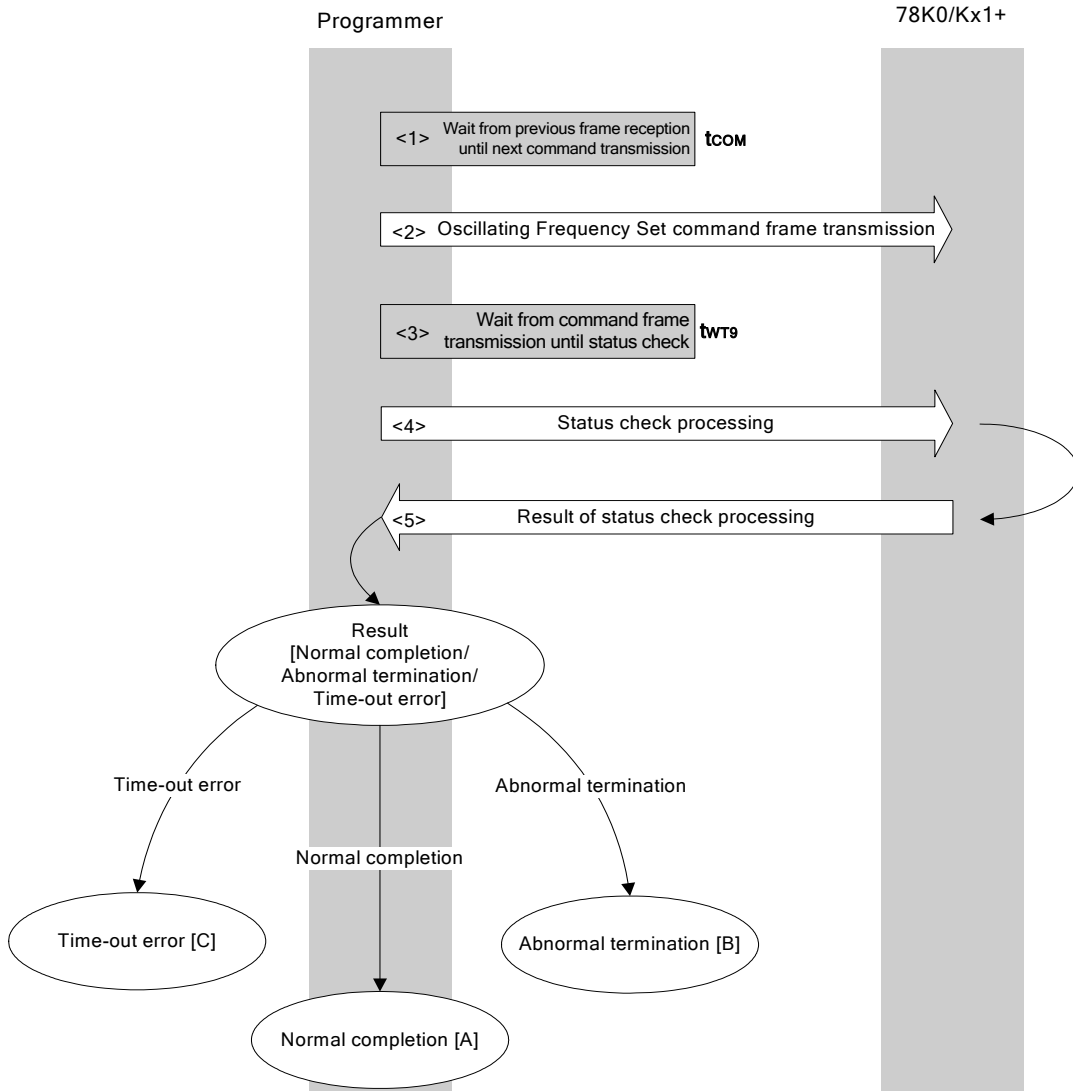
        if (rc == FLC_DFTO_ERR)        // timeout error ?
            break;                    // yes // case [C]
        if (rc == FLC_ACK)            // Ack ?
            break;                    // yes // case [A]
        //continue;                    // case [B] (if exit from loop)
    }
    // switch(rc) {
    //
    //     case  FLC_NO_ERR:  return rc;  break; // case [A]
    //     case  FLC_DFTO_ERR: return rc;  break; // case [C]
    //     default:          return rc;  break; // case [B]
    // }
    return rc;
}

```

8.6 Oscillating Frequency Set Command

8.6.1 Processing sequence chart

Oscillating Frequency Set command processing sequence



8.6.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Oscillating Frequency Set command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WT9} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]

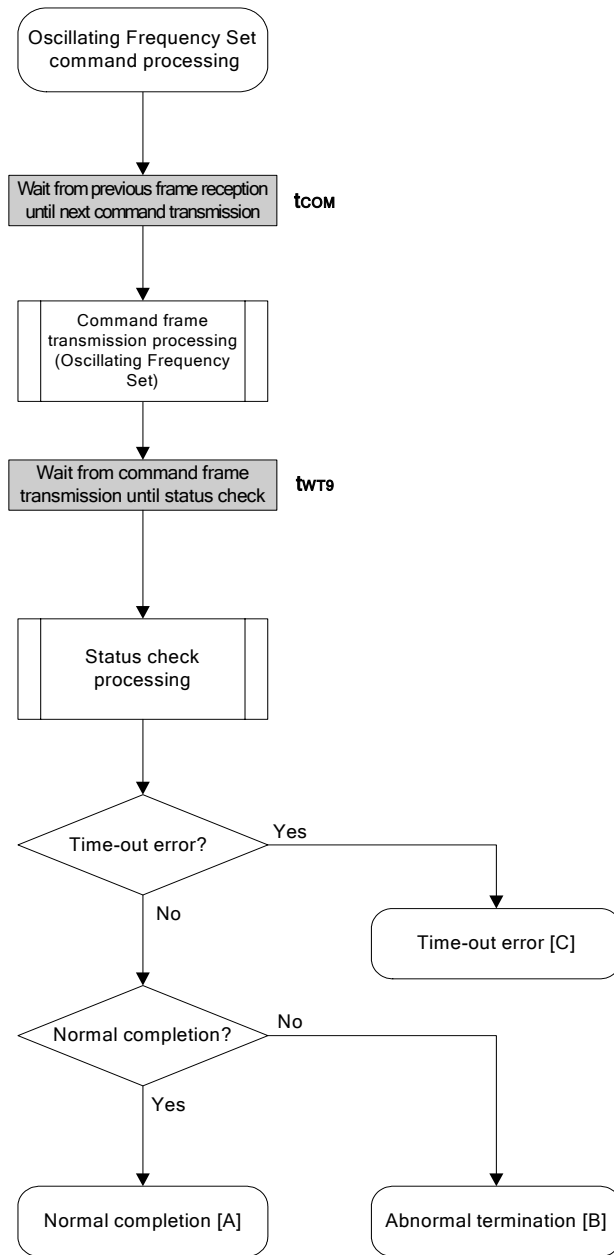
When the processing ends abnormally: Abnormal termination [B]

When a time-out error occurs: A time-out error [C] is returned.

8.6.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the operating frequency was correctly set to the 78K0/Kx1+.
Abnormal termination [B]	Parameter error	05H	The oscillation frequency value is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.

8.6.4 Flowchart



8.6.5 Sample program

The following shows a sample program for Oscillating Frequency Set command processing.

```

/*****
/*
/* Set Flash device clock value command (CSI)
/*
/*
/*****
/* [i] u8 clk[4] ... frequency data(D1-D4)
/* [r] u16 ... error code
/*****
u16 fl_csi_setclk(u8 clk[])
{
    u16 rc;

    fl_cmd_prm[0] = clk[0]; // "D01"
    fl_cmd_prm[1] = clk[1]; // "D02"
    fl_cmd_prm[2] = clk[2]; // "D03"
    fl_cmd_prm[3] = clk[3]; // "D04"

    fl_wait(tCOM); // wait before sending command frame

    put_cmd_csi(FL_COM_SET_OSC_FREQ, 5, fl_cmd_prm);
    // send "Oscillation Frequency Set" command

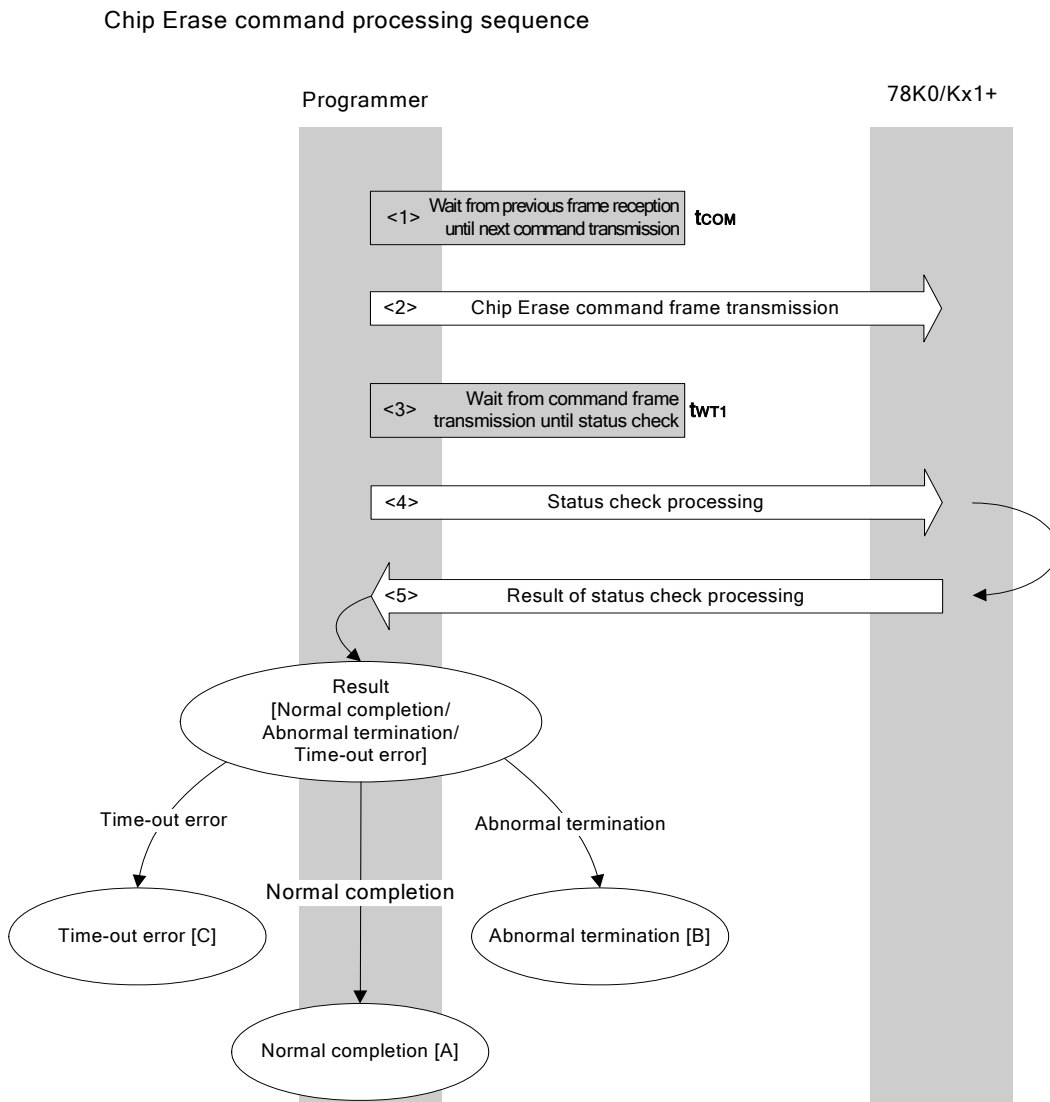
    fl_wait(tWT9);

    rc = fl_csi_getstatus(); // get status frame
    // switch(rc) {
    //
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}

```

8.7 Chip Erase Command

8.7.1 Processing sequence chart



8.7.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Chip Erase command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WT1 (MAX.)}$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]

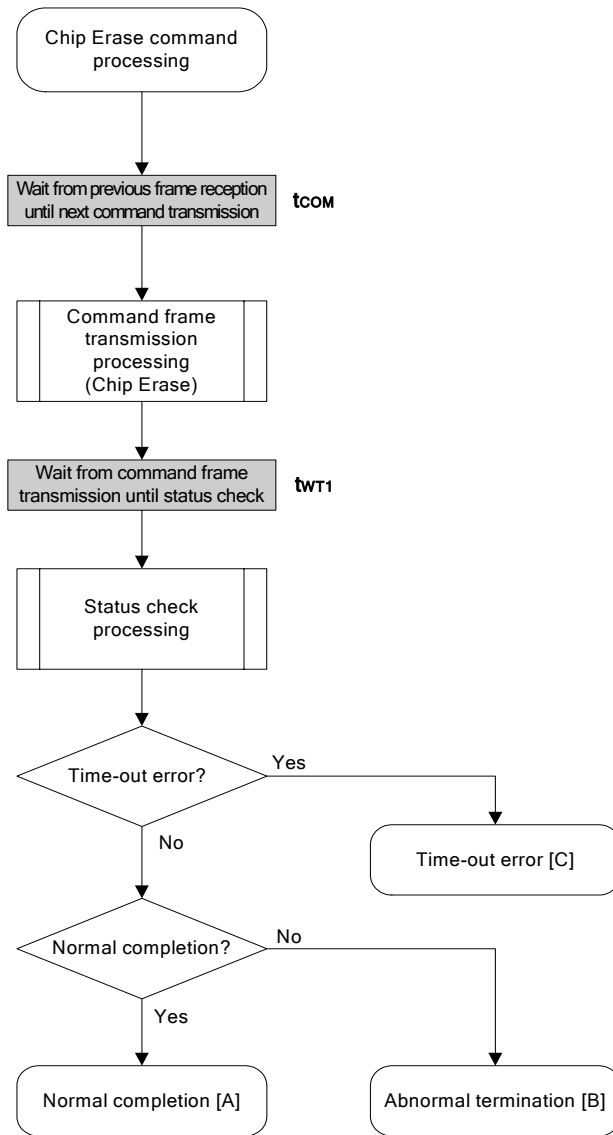
When the processing ends abnormally: Abnormal termination [B]

When a time-out error occurs: A time-out error [C] is returned.

8.7.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and chip erase was performed normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Chip erase is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Erase error	1AH	An erase error has occurred.
Time-out error [C]		–	The status frame was not received within the specified time.

8.7.4 Flowchart



8.7.5 Sample program

The following shows a sample program for Chip Erase command processing.

```

/*****
/*
/* Erase all(chip) command (CSI)
/*
/*****
/* [r] u16      ... error code
/*****
u16      fl_csi_erase_all(void)
{
    u16    rc;

    fl_wait(tCOM);                // wait before sending command frame

    put_cmd_csi(FL_COM_ERASE_CHIP, 1, fl_cmd_prm); // send "Chip Erase" command

    fl_wait(tWT1);

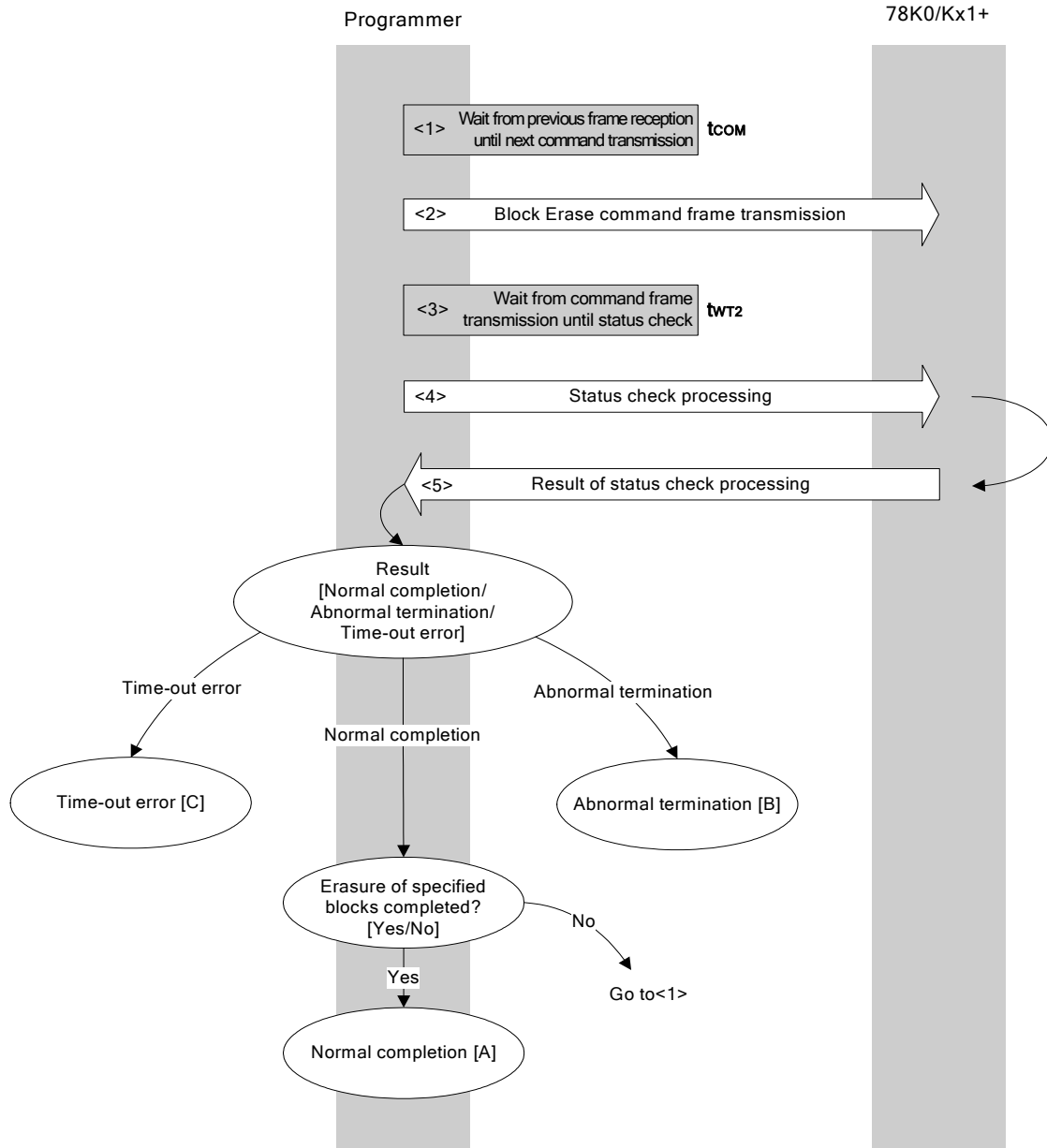
    rc = fl_csi_getstatus(); // get status frame
    // switch(rc) {
    //
    //     case  FLC_NO_ERR:  return rc;    break; // case [A]
    //     case  FLC_DFTO_ERR: return rc;    break; // case [C]
    //     default:          return rc;    break; // case [B]
    // }
    return rc;
}

```

8.8 Block Erase Command

8.8.1 Processing sequence chart

Block Erase command processing sequence



8.8.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Block Erase command is transmitted by command frame transmission processing.
- <3> Waits until status frame acquisition (wait time $t_{WT2} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: When the block erase for all of the specified blocks is not yet completed, processing changes the block number and re-executes the sequence from <1>.

When the block erase for all of the specified blocks is completed, the processing ends normally [A].

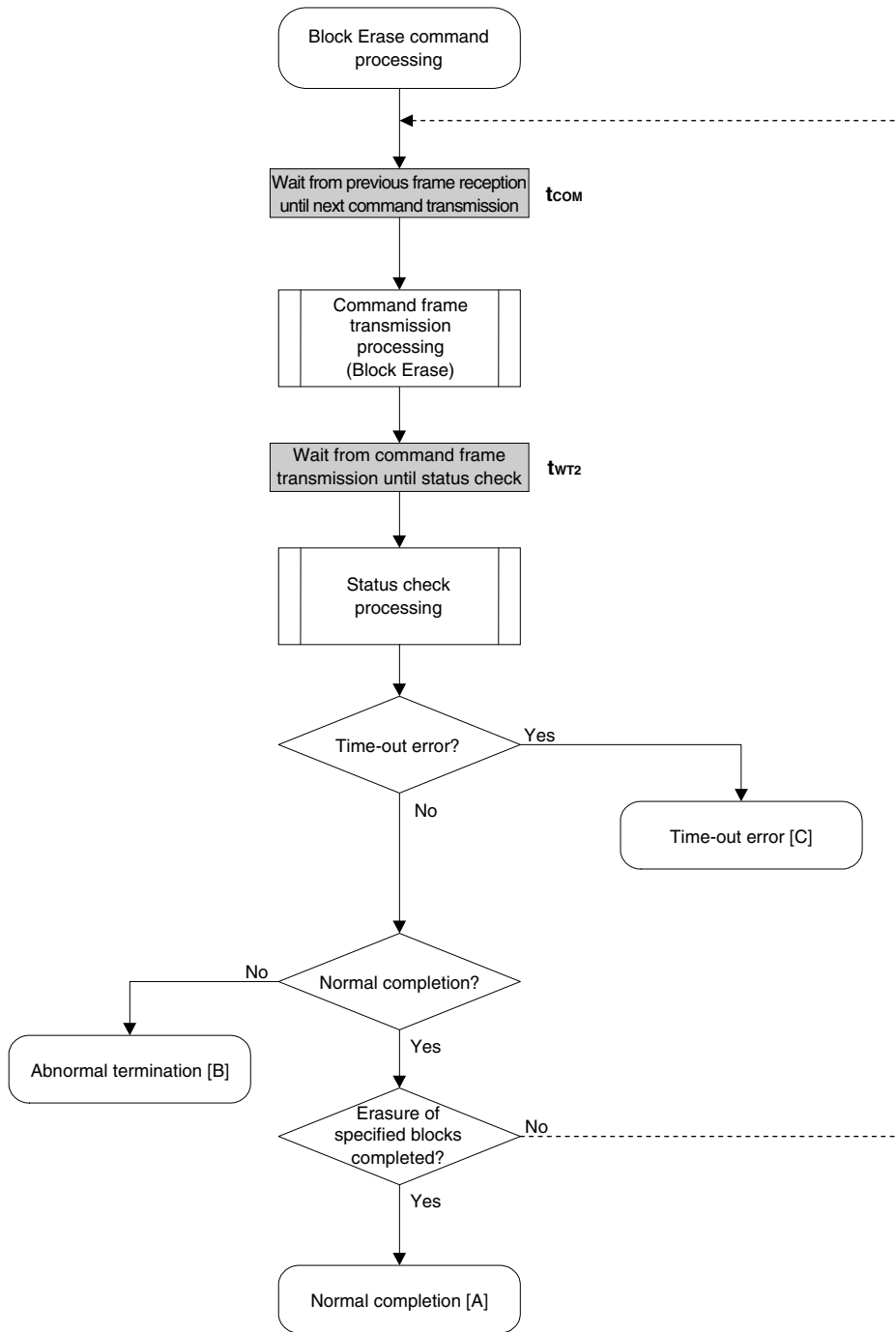
When the processing ends abnormally: Abnormal termination [B]

When a time-out error occurs: A time-out error [C] is returned.

8.8.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and block erase was performed normally.
Abnormal termination [B]	Parameter error	05H	The block number is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Chip erase is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	Erase error	1AH	An erase error has occurred.
Time-out error [C]		–	The status frame was not received within the specified time.

8.8.4 Flowchart



8.8.5 Sample program

The following shows a sample program for Block Erase command processing for one block.

```

/*****
/*
/* Erase block command (CSI)
/*
/*
/*****
/* [i] u8 block    ... block number
/* [r] u16         ... error code
/*****
u16    fl_csi_erase_blk(u8 block)
{
    u16    rc;

    fl_cmd_prm[0] = block;    // set params

    fl_wait(tCOM);           // wait before sending command frame

    put_cmd_csi(FL_COM_ERASE_BLOCK, 2, fl_cmd_prm);    // send "Block Erase" command

    fl_wait(tWT2);

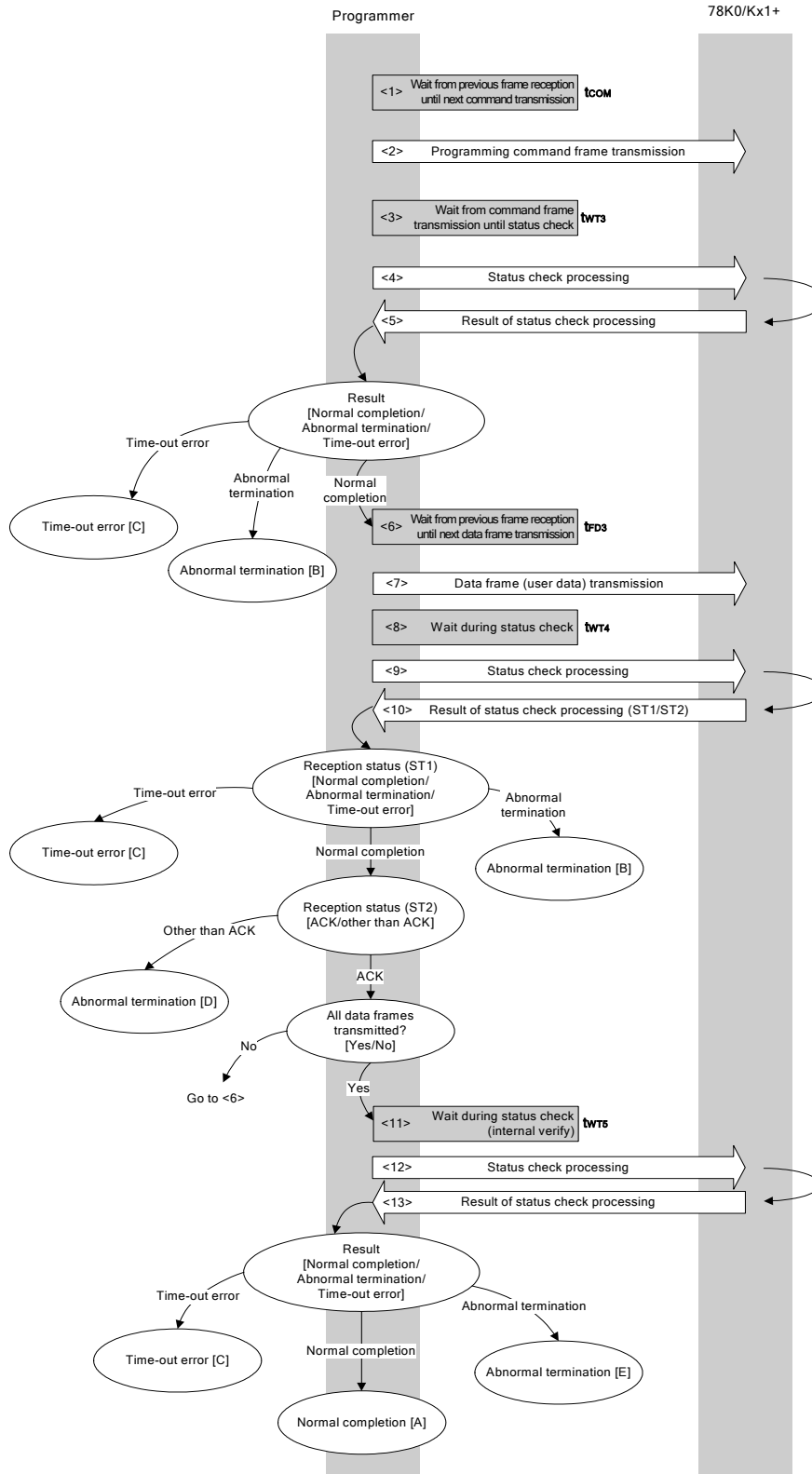
    rc = fl_csi_getstatus(); // get status frame
    // switch(rc) {
    //
    //     case  FLC_NO_ERR:  return rc;    break; // case [A]
    //     case  FLC_DFTO_ERR: return rc;    break; // case [C]
    //     default:          return rc;    break; // case [B]
    // }
    return rc;
}

```

8.9 Programming Command

8.9.1 Processing sequence chart

Programming command processing sequence



8.9.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Programming command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time t_{WT3} (MAX.)).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.

When the processing ends abnormally: Abnormal termination [B]

When a time-out error occurs: A time-out error [C] is returned.

- <6> Waits until the next data frame transmission (wait time t_{FD3} (MAX.)).
- <7> User data to be written to the 78K0/Kx1+ flash memory is transmitted by data frame transmission processing.
- <8> Waits from data frame (user data) transmission until status check processing (wait time t_{WT4} (MAX.)).
- <9> The status frame is acquired by status check processing.
- <10> The following processing is performed according to the result of status check processing (status code (ST1/ST2)) (also refer to the processing sequence chart and flowchart).

When ST1 = abnormal termination: Abnormal termination [B]

When ST1 = time-out error: A time-out error [C] is returned.

When ST1 = normal completion: The following processing is performed according to the ST2 value.

- When ST2 \neq ACK: Abnormal termination [D]
- When ST2 = ACK: Proceeds to <11> when transmission of all of the user data is completed.
If there still remain user data to be transmitted, the processing re-executes the sequence from <6>.

<11> Waits until status check processing (time-out time t_{WT5} (MAX.)).

<12> The status frame is acquired by status check processing.

<13> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]

(Indicating that the internal verify check has performed normally after completion of write)

When the processing ends abnormally: Abnormal termination [E]

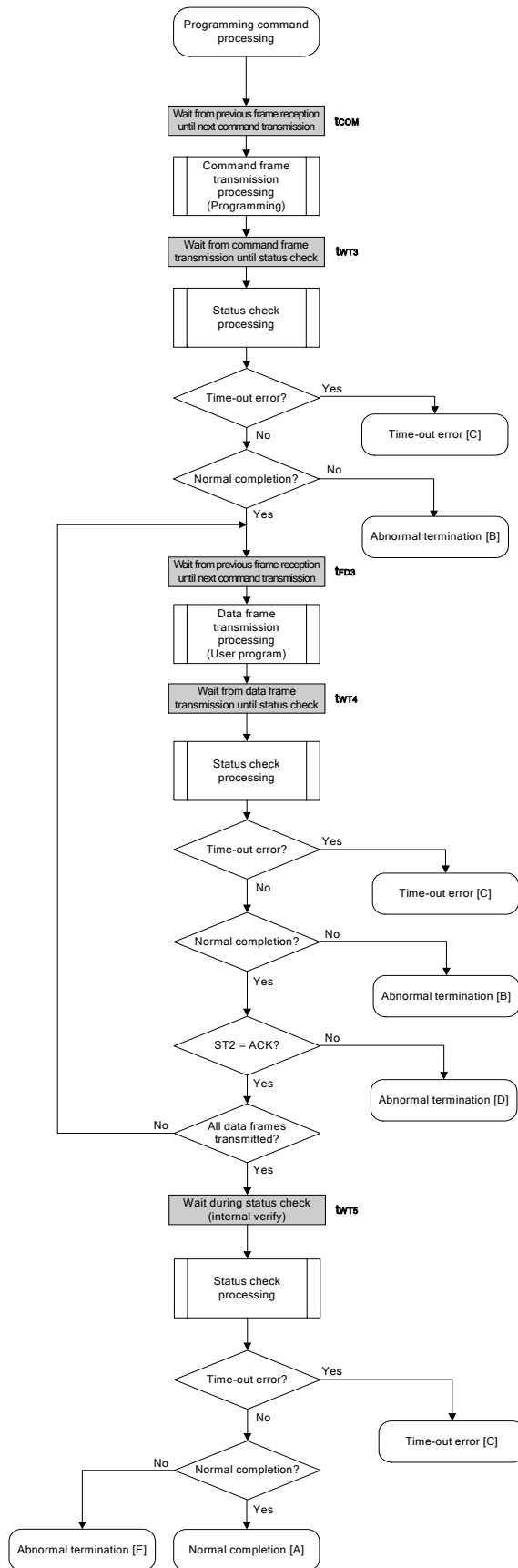
(Indicating that the internal verify check has not performed normally after completion of write)

When a time-out error occurs: A time-out error [C] is returned.

8.9.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the user data was written normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or is not a multiple of 8.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Protect error	10H	Write is prohibited in the security setting.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.
Abnormal termination [D]	FLMD error	18H (ST2)	A write error has occurred.
	Write error	1CH (ST2)	A write error has occurred.
Abnormal termination [E]	MRG11 error	1BH	An internal verify error has occurred.

8.9.4 Flowchart



8.9.5 Sample program

The following shows a sample program for Programming command processing.

```

/*****
/*
/* Write command (CSI)
/*
/*****
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****
u16      fl_csi_write(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;
    u16    block_num;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    block_num = get_block_num(top, bottom); // get block num

    /*****
    /*      send command & check status
    /*****

    fl_wait(tCOM);
    put_cmd_csi(FL_COM_WRITE, 7, fl_cmd_prm); // send "Programming" command
    fl_wait(tWT3);

    rc = fl_csi_getstatus(); // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_DFTO_ERR:            return rc; break; // case [C]
        default:                          return rc; break; // case [B]
    }

    /*****
    /*      send user data
    /*****

    send_head = top;

    while(1){

        if ((bottom - send_head) > 256){ // rest size > 256 ?
            is_end = false; // yes, not end frame
            send_size = 256; // transmit size = 256 byte
        }
        else{
            is_end = true;

```



```

        send_size = bottom - send_head + 1;
                        // transmit size = (bottom - send_head)+1 byte
    }

    memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                        // set data frame payload
    send_head += send_size;

    fl_wait(tFD3);                // wait before sending data frame
    put_dfrm_csi(send_size, fl_txdata_frm, is_end);
                                    // send data frame (user data)
    fl_wait(tWT4);                // wait

    rc = fl_csi_getstatus(); // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
    // case FLC_DFTO_ERR: return rc;    break; // case [C]
        default:                return rc;    break; // case [B]
    }
    if (fl_st2 != FLST_ACK){                // ST2 = ACK ?
        rc = decode_status(fl_st2);        // No
        return rc;                        // case [D]
    }

    if (is_end)                // send all user data ?
        break;                    // yes
    //continue;

}
/*****
/*    Check internally verify    */
*****/

    fl_wait(tWT5 * block_num);                // wait

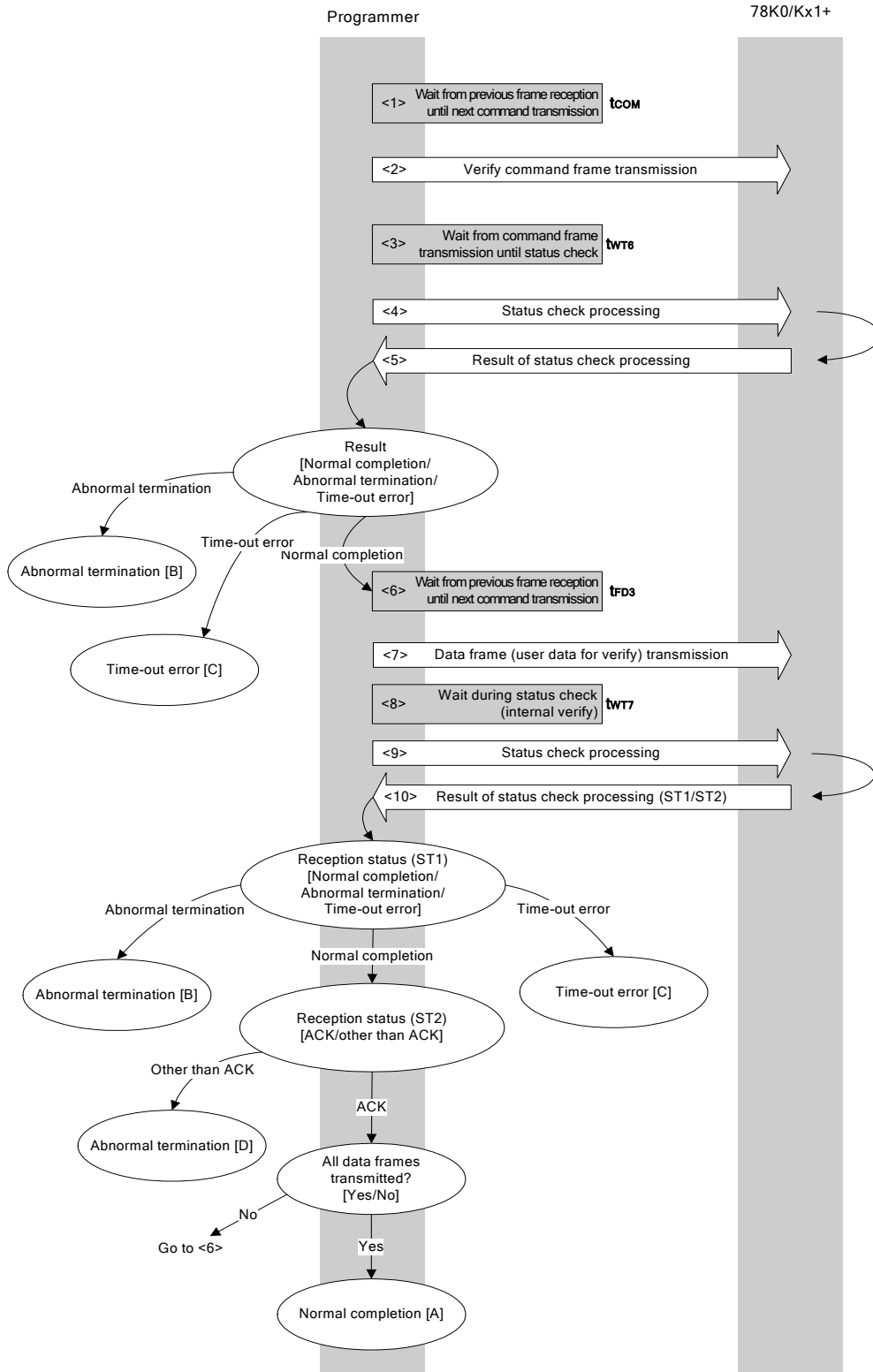
    rc = fl_csi_getstatus();                // get status frame
// switch(rc) {
//     case FLC_NO_ERR:    return rc;    break; // case [A]
//     case FLC_DFTO_ERR: return rc;    break; // case [C]
//     default:                return rc;    break; // case [E]
// }
    return rc;
}

```

8.10 Verify Command

8.10.1 Processing sequence chart

Verify command processing sequence



8.10.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Verify command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WT6} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

- <6> Waits from the previous frame reception until the next data frame transmission (wait time t_{FD3}).
- <7> User data for verifying is transmitted by data frame transmission processing.
- <8> Waits from data frame transmission until status check processing (wait time $t_{WT7} (MAX.)$).
- <9> The status frame is acquired by status check processing.
- <10> The following processing is performed according to the result of status check processing (status code (ST1/ST2)) (also refer to the processing sequence chart and flowchart).

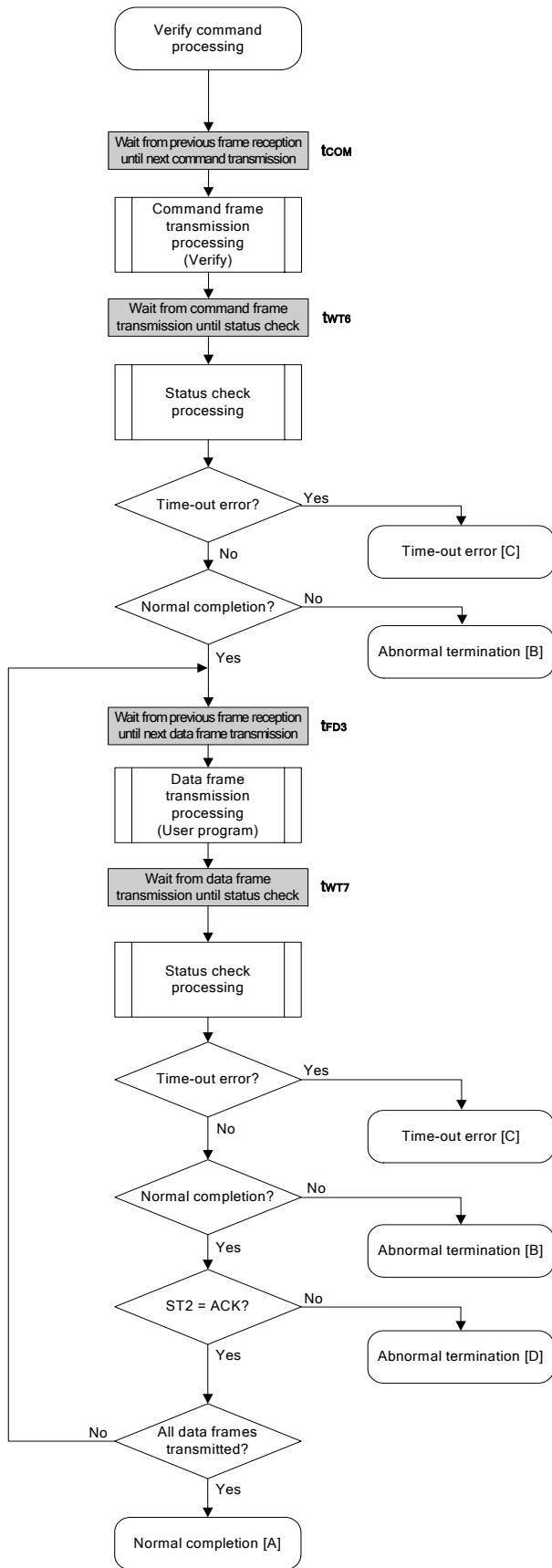
When ST1 = abnormal termination: Abnormal termination [B]
 When ST1 = time-out error: A time-out error [C] is returned.
 When ST1 = normal completion: The following processing is performed according to the ST2 value.

- When ST2 \neq ACK: Abnormal termination [D]
- When ST2 = ACK: If transmission of all data frames is completed, the processing ends normally [A].
 If there still remain data frames to be transmitted, the processing re-executes the sequence from <6>.

8.10.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the verify was completed normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame or data frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		-	The status frame was not received within the specified time.
Abnormal termination [D]	Verify error	0FH (ST2)	The verify has failed, or another error has occurred.

8.10.4 Flowchart



8.10.5 Sample program

The following shows a sample program for Verify command processing.

```

/*****
/*
/* Verify command (CSI)
/*
/*****
/* [i] u32 top      ... start address
/* [i] u32 bottom  ... end address
/* [r] u16         ... error code
/*****
u16      fl_csi_verify(u32 top, u32 bottom)
{
    u16    rc;
    u32    send_head, send_size;
    bool   is_end;

    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    /*****
    /*      send command & check status
    /*****
    fl_wait(tCOM);
    put_cmd_csi(FL_COM_VERIFY, 7, fl_cmd_prm); // send "Verify" command
    fl_wait(tWT6);

    rc = fl_csi_getstatus(); // get status frame
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_DFTO_ERR:      return rc; break; // case [C]
        default:                  return rc; break; // case [B]
    }

    /*****
    /*      send user data
    /*****
    send_head = top;

    while(1){
        if ((bottom - send_head) > 256){ // rest size > 256 ?
            is_end = false; // yes, not end frame
            send_size = 256; // transmit size = 256 byte
        }
    }
}

```

```
else{
    is_end = true;
    send_size = bottom - send_head + 1;
                // transmit size = (bottom - send_head)+1 byte
}

memcpy(fl_txdata_frm, rom_buf+send_head, send_size);
                // set data frame payload
send_head += send_size;

fl_wait(tFD3);                // wait before sending data frame
put_dfrm_csi(send_size, fl_txdata_frm, is_end);        // send data frame
fl_wait(tWT7);                // wait

rc = fl_csi_getstatus(); // get status frame
switch(rc) {
    case FLC_NO_ERR:                break; // continue
// case FLC_DFTO_ERR: return rc;    break; // case [C]
    default:                return rc;    break; // case [B]
}
if (fl_st2 != FLST_ACK){                // ST2 = ACK ?
    rc = decode_status(fl_st2);        // No
    return rc;                // case [D]
}

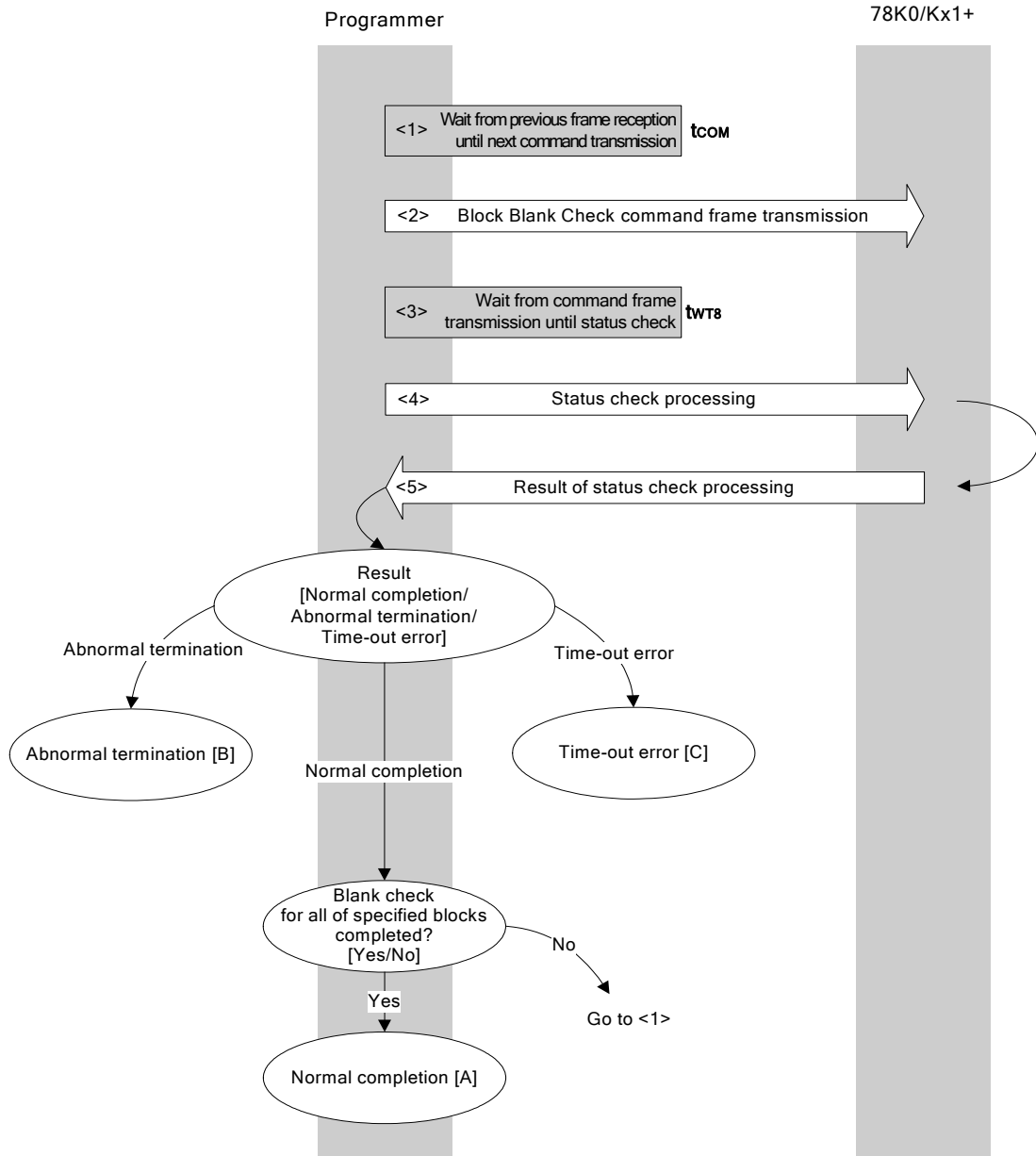
if (is_end)                // send all user data ?
    break;                // yes
//continue;

}
return FLC_NO_ERR; // case [A]
}
```

8.11 Block Blank Check Command

8.11.1 Processing sequence chart

Block Blank Check command processing sequence



8.11.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Block Blank Check command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WTS} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When a time-out error occurs: A time-out error [C] is returned.

When the processing ends abnormally: Abnormal termination [B]

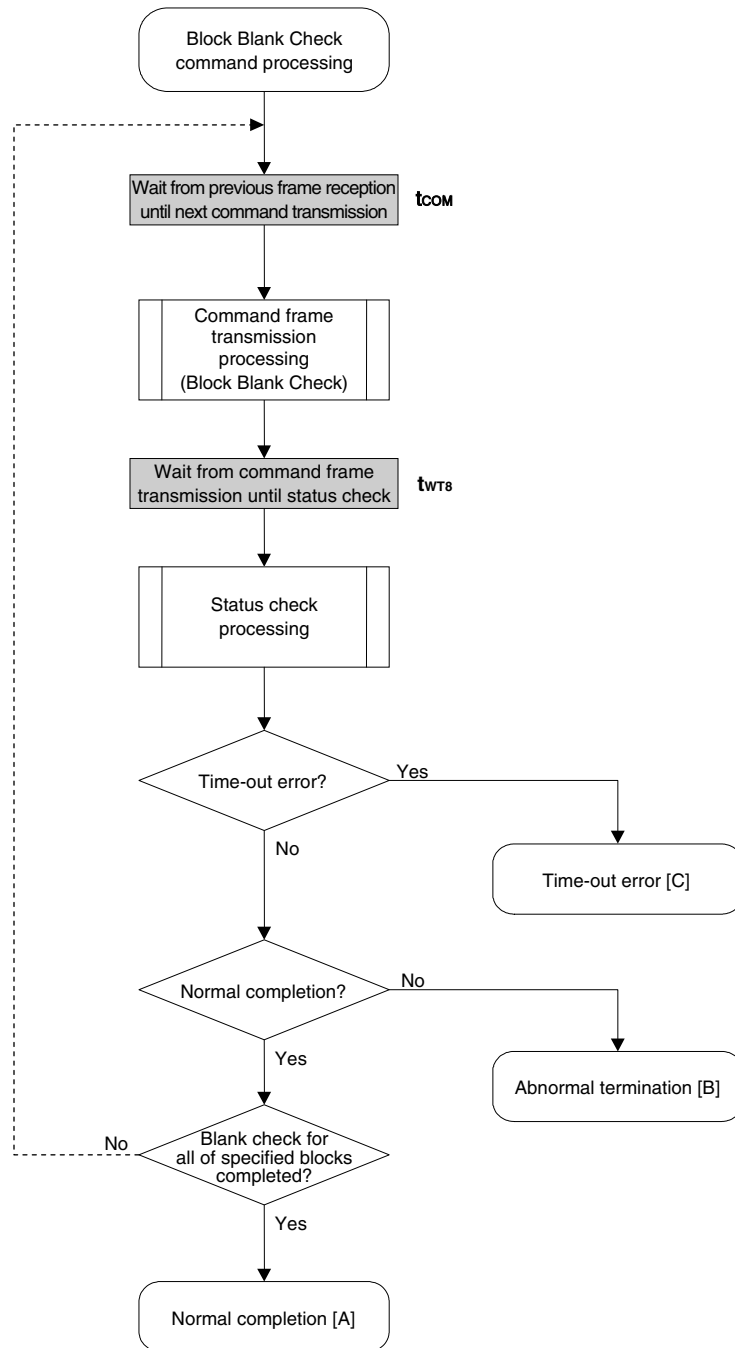
When the processing ends normally: If the blank check for all of the specified blocks is not yet completed, processing changes the block number and re-executes the sequence from <1>.

If the blank check for all of the specified blocks is completed, the processing ends normally [A].

8.11.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and all of the specified blocks are blank.
Abnormal termination [B]	Parameter error	05H	The block number is out of range.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
	MRG11 error	1BH	The specified block in the flash memory is not blank.
Time-out error [C]		–	The status frame was not received within the specified time.

8.11.4 Flowchart



8.11.5 Sample program

The following shows a sample program for Block Blank Check command processing for one block.

```

/*****
/*
/* Block blank check command (CSI)
/*
/*****
/* [i] u8 block ... block number
/* [r] u16 ... error code
/*****
u16 fl_csi_blk_blank_chk(u8 block)
{
    u16 rc;

    fl_cmd_prm[0] = block; // "BLK"

    fl_wait(tCOM); // wait before sending command frame

    put_cmd_csi(FL_COM_BLOCK_BLANK_CHK, 2, fl_cmd_prm);
    // send "Block Blank Check" command

    fl_wait(tWT8);

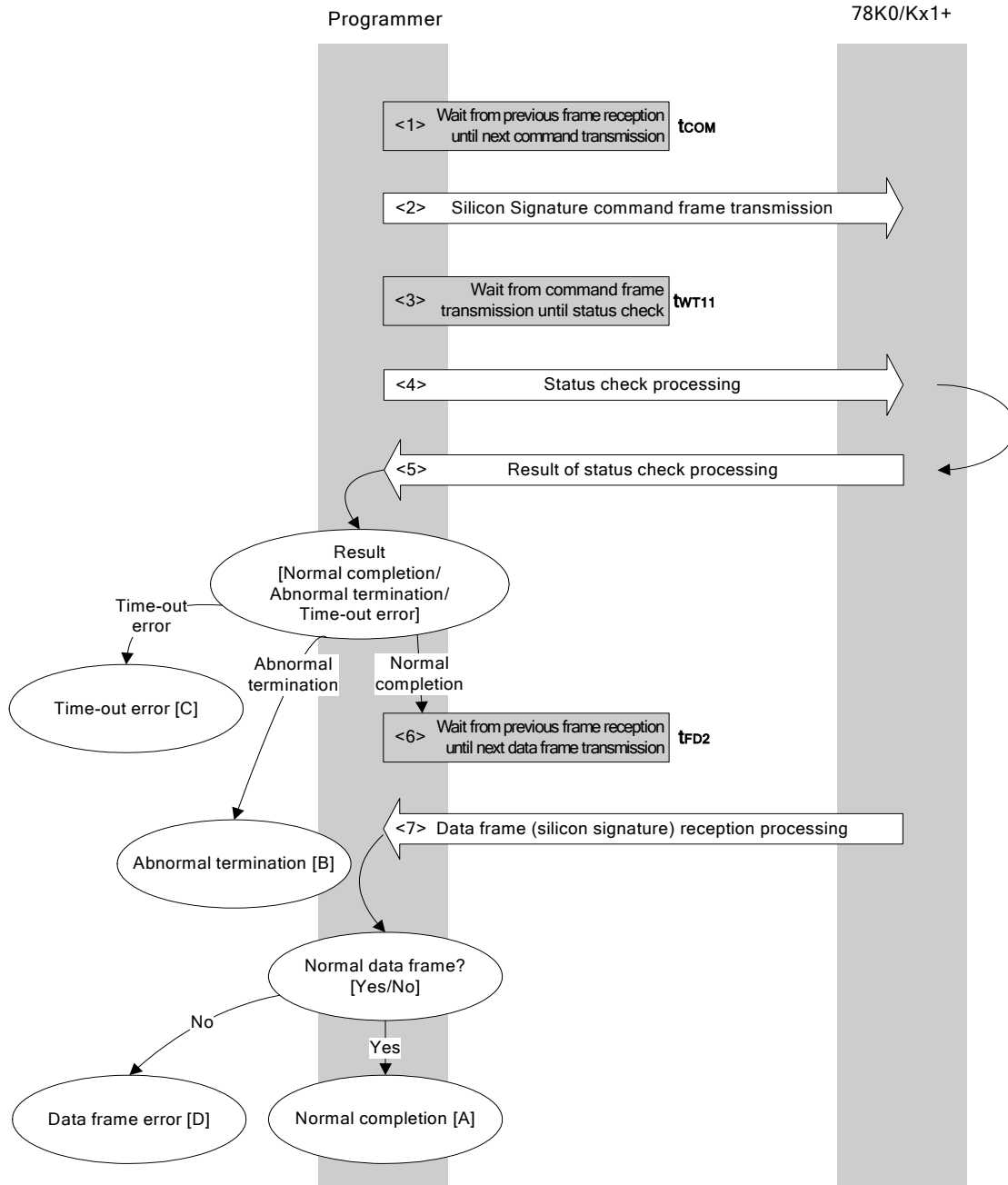
    rc = fl_csi_getstatus(); // get status frame
    // switch(rc) {
    //
    //     case FLC_NO_ERR: return rc; break; // case [A]
    //     case FLC_DFTO_ERR: return rc; break; // case [C]
    //     default: return rc; break; // case [B]
    // }
    return rc;
}

```

8.12 Silicon Signature Command

8.12.1 Processing sequence chart

Silicon Signature command processing sequence



8.12.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Silicon Signature command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WT11} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

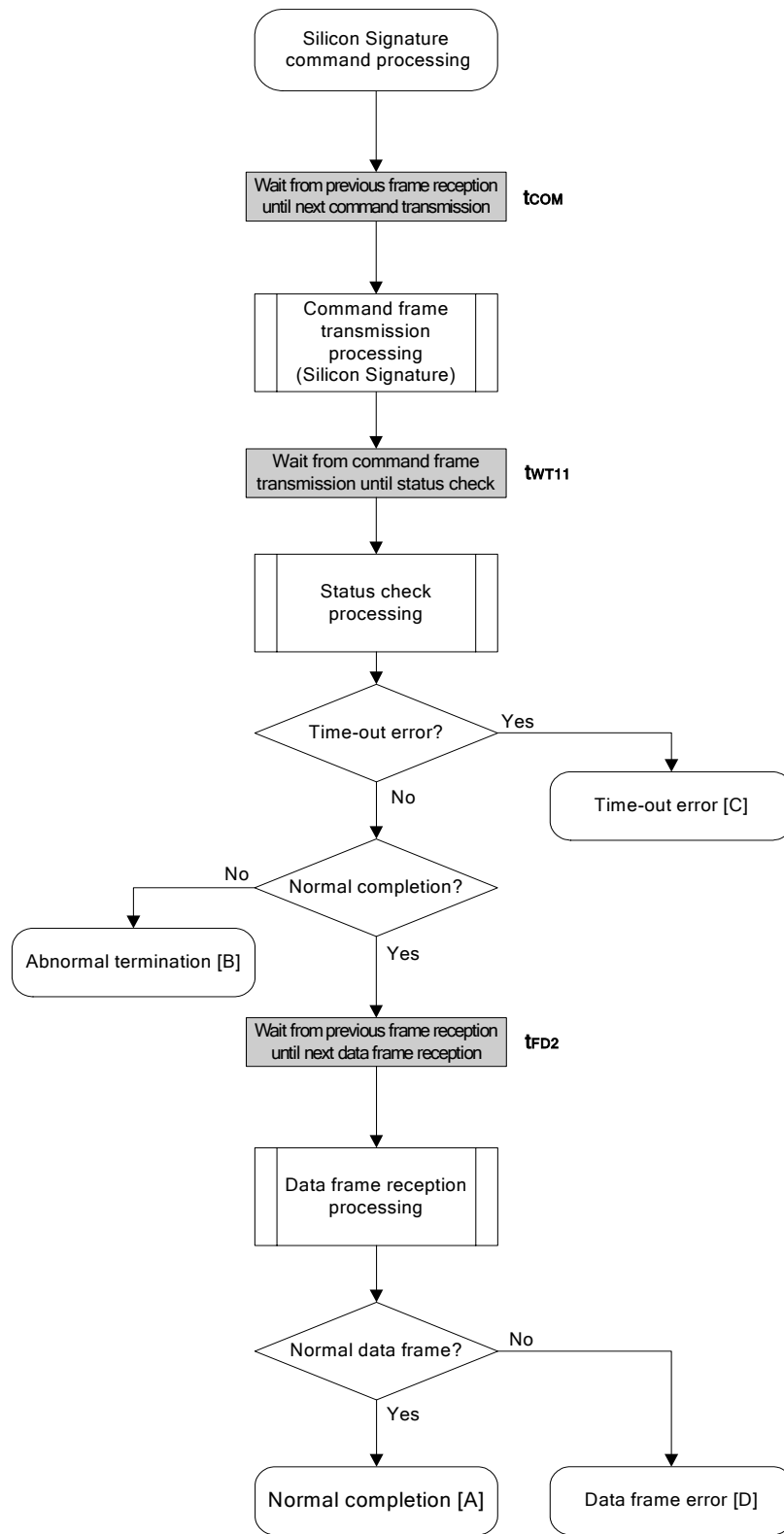
- <6> Waits from the previous frame reception until the next command transmission (wait time t_{FD2}).
- <7> The received data frame (silicon signature data) is checked.

If data frame is normal: Normal completion [A]
 If data frame is abnormal: Data frame error [D]

8.12.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and the silicon signature was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as silicon signature data does not match.

8.12.4 Flowchart



8.12.5 Sample program

The following shows a sample program for Silicon Signature command processing.

```

/*****
/*
/* Get silicon signature command (CSI)
/*
/*****
/* [i] u8 *sig    ... pointer to signature save area
/* [r] u16        ... error code
/*****
u16    fl_csi_getsig(u8 *sig)
{
    u16    rc;

    fl_wait(tCOM);                // wait before sending command frame

    put_cmd_csi(FL_COM_GET_SIGNATURE, 1, fl_cmd_prm);
                                // send "Silicon Signature" command

    fl_wait(tWT11);

    rc = fl_csi_getstatus();    // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_DFTO_ERR:            return rc;    break; // case [C]
        default:                        return rc;    break; // case [B]
    }

    fl_wait(tFD2_SIG);            // wait before getting data frame

    rc = get_dfrm_csi(fl_rxdata_frm);    // get data frame (signature data)

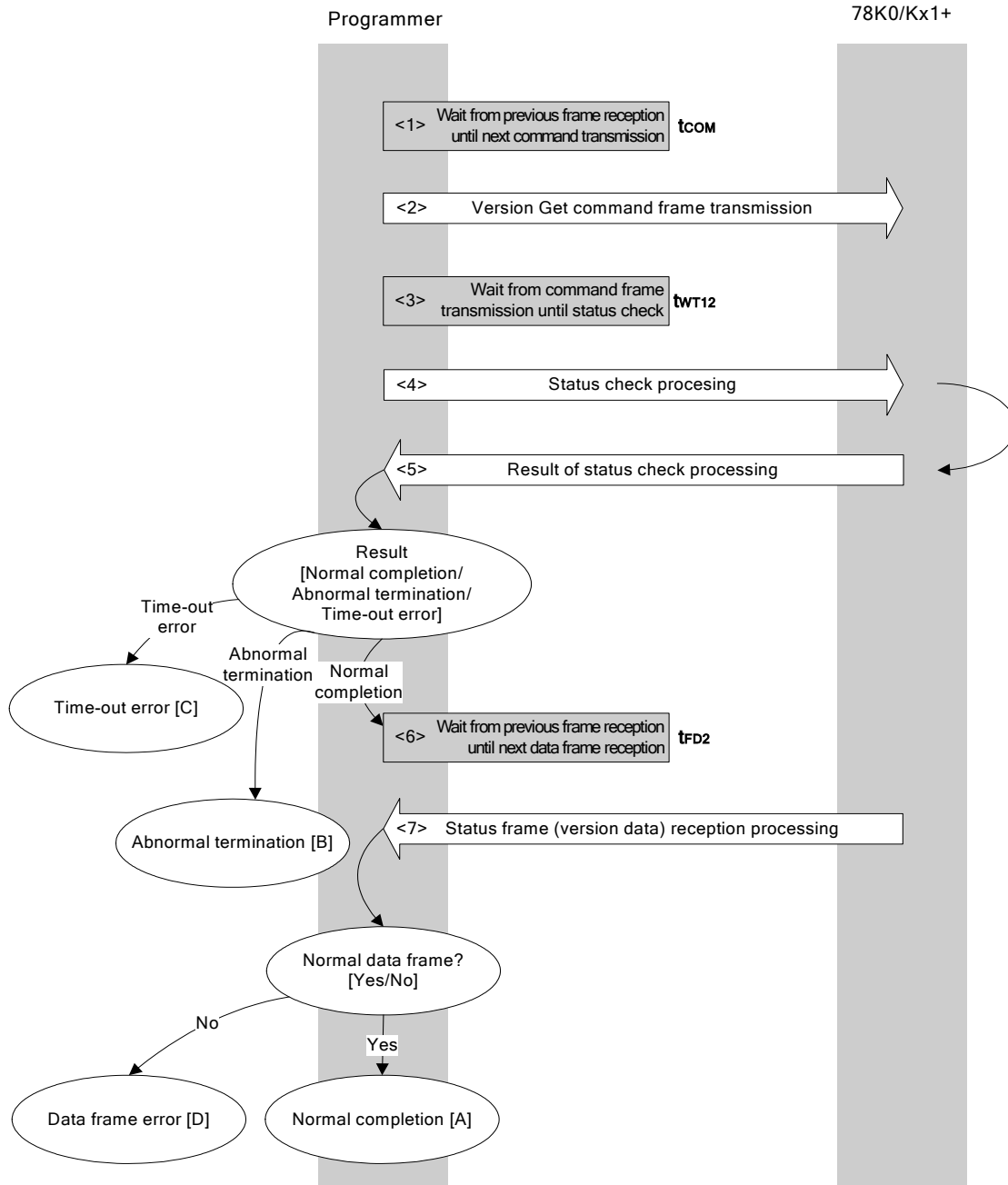
    if (rc){                        // if no error,
        return rc;                    // case [D]
    }
    memcpy(sig, fl_rxdata_frm+OFS_STA_PLD, fl_rxdata_frm[OFS_LEN]);
                                // copy Signature data
    return rc;                        // case [A]
}

```

8.13 Version Get Command

8.13.1 Processing sequence chart

Version Get command processing sequence



8.13.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Version Get command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WT12} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.

When the processing ends abnormally: Abnormal termination [B]

When a time-out error occurs: A time-out error [C] is returned.

- <6> Waits from the previous frame reception until the next command transmission (wait time t_{FD2}).

- <7> The received data frame (version data) is checked.

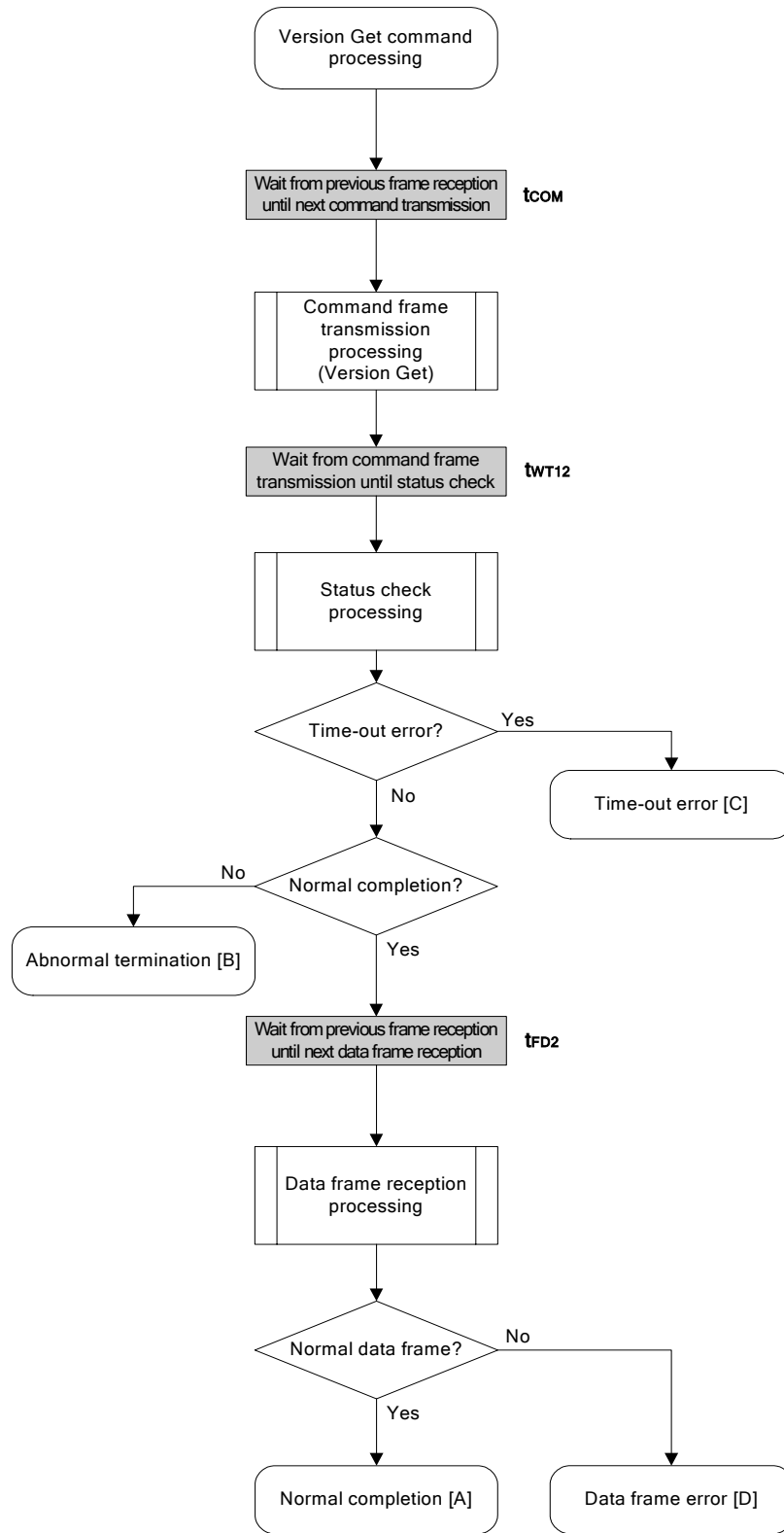
If data frame is normal: Normal completion [A]

If data frame is abnormal: Data frame error [D]

8.13.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and version data was acquired normally.
Abnormal termination [B]	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as version data does not match.

8.13.4 Flowchart



8.13.5 Sample program

The following shows a sample program for Version Get command processing.

```

/*****
/*
/* Get device/firmware version command (CSI)
/*
/*****
/* [i] u8 *buf    ... pointer to version data save area
/* [r] u16       ... error code
/*****
u16    fl_csi_getver(u8 *buf)
{
    u16    rc;

    fl_wait(tCOM);                // wait before sending command frame

    put_cmd_csi(FL_COM_GET_VERSION, 1, fl_cmd_prm);    // send "Version Get" command

    fl_wait(tWT12);

    rc = fl_csi_getstatus();    // get status frame
    switch(rc) {
        case FLC_NO_ERR:                break; // continue
        // case FLC_DFTO_ERR:    return rc;    break; // case [C]
        default:                return rc;    break; // case [B]
    }

    fl_wait(tFD2_VG);                // wait before getting data frame

    rc = get_dfrm_csi(fl_rxd_data_frm);    // get version data

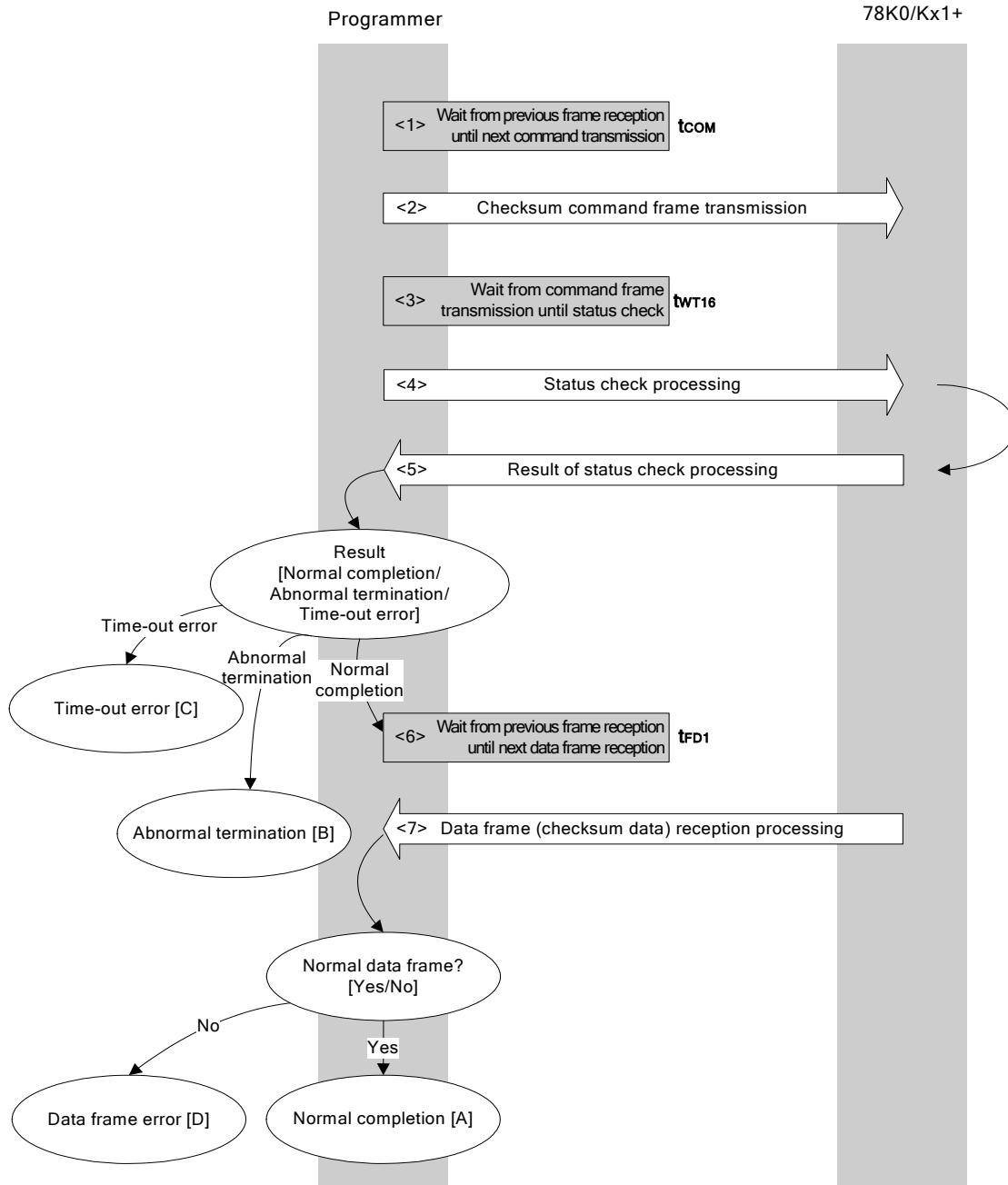
    if (rc){                            // if no error,
        return rc;                        // case [D]
    }
    memcpy(buf, fl_rxd_data_frm+OFS_STA_PLD, DFV_LEN); // copy version data
    return rc;                            // case [A]
}

```

8.14 Checksum Command

8.14.1 Processing sequence chart

Checksum command processing sequence



8.14.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Checksum command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WT16} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

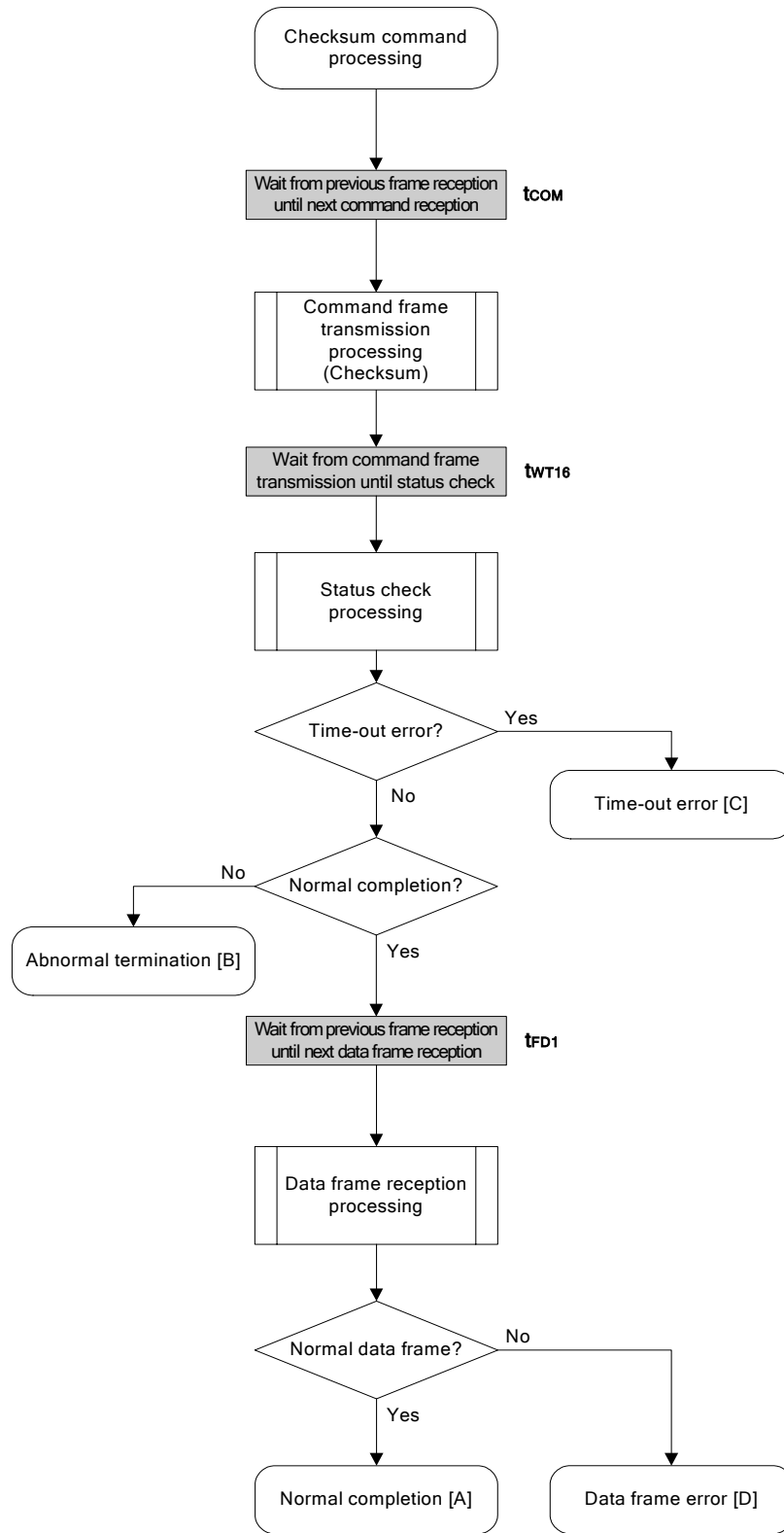
- <6> Waits from the previous frame reception until the next command transmission (wait time t_{FD1}).
- <7> The received data frame (checksum data) is checked.

If data frame is normal: Normal completion [A]
 If data frame is abnormal: Data frame error [D]

8.14.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and checksum data was acquired normally.
Abnormal termination [B]	Parameter error	05H	The specified start/end address is out of the flash memory range, or the specified address is not a fixed address in 2 KB units.
	Checksum error	07H	The checksum of the transmitted command frame does not match.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.
Data frame error [D]		–	The checksum of the data frame received as version data does not match.

8.14.4 Flowchart



8.14.5 Sample program

The following shows a sample program for Checksum command processing.

```

/*****/
/*
/* Get checksum command (CSI)
/*
/*****/
/* [i] u16 *sum    ... pointer to checksum save area
/* [i] u32 top    ... start address
/* [i] u32 bottom ... end address
/* [r] u16        ... error code
/*****/
u16    fl_csi_getsum(u16 *sum, u32 top, u32 bottom)
{
    u16    rc;
    u16    block_num;

    /*****/
    /*    set params
    /*****/
    // set params
    set_range_prm(fl_cmd_prm, top, bottom); // set SAH/SAM/SAL, EAH/EAM/EAL

    block_num = get_block_num(top, bottom); // get block num

    /*****/
    /*    send command
    /*****/
    fl_wait(tCOM); // wait before sending command frame

    put_cmd_csi(FL_COM_GET_CHECK_SUM, 7, fl_cmd_prm); // send "Checksum" command

    fl_wait(tWT16);

    rc = fl_csi_getstatus(); // get status frame
    switch(rc) {
        case FLC_NO_ERR:          break; // continue
        // case FLC_DFTO_ERR: return rc; break; // case [C]
        default:                  return rc; break; // case [B]
    }

    /*****/
    /*    get data frame (Checksum data)
    /*****/
    fl_wait(tFD1 * block_num); // wait before getting data frame

```

```
rc = get_dfrm_csi(fl_rxddata_frm); // get data frame(version data)

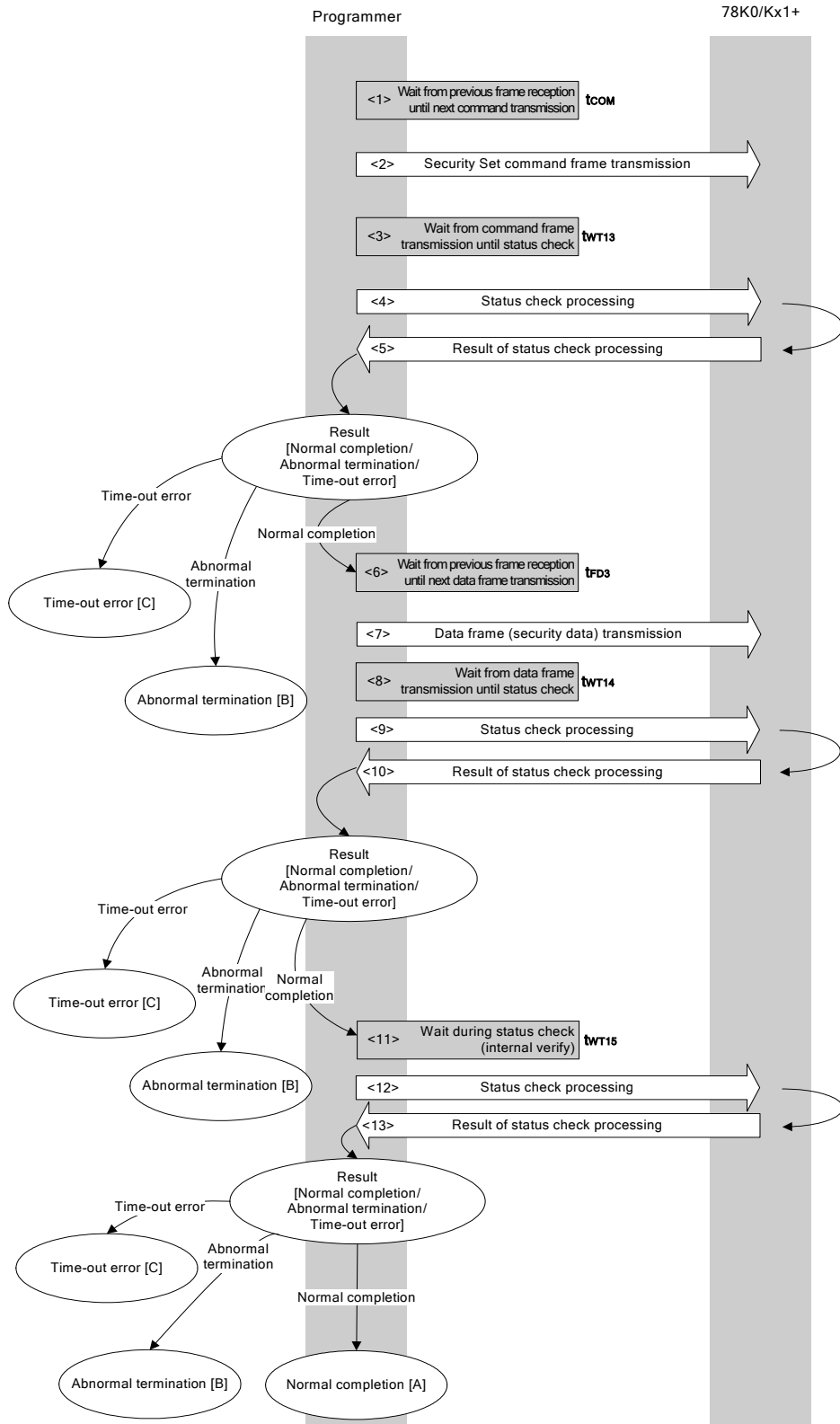
if (rc){
    return rc; // if error, // case [D]
}

*sum = (fl_rxddata_frm[OFS_STA_PLD] << 8) + fl_rxddata_frm[OFS_STA_PLD+1]; // set SUM data
return rc; // case [A]
}
```

8.15 Security Set Command

8.15.1 Processing sequence chart

Security Set command processing sequence



8.15.2 Description of processing sequence

- <1> Waits from the previous frame reception until the next command transmission (wait time t_{COM}).
- <2> The Security Set command is transmitted by command frame transmission processing.
- <3> Waits from command transmission until status check processing (wait time $t_{WT13} (MAX.)$).
- <4> The status frame is acquired by status check processing.
- <5> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <6>.
 When the processing ends abnormally: Abnormal termination [B]
 When a time-out error occurs: A time-out error [C] is returned.

- <6> Waits from the previous frame reception until the data frame transmission (wait time t_{FD3}).
- <7> The data frame (security setting data) is transmitted by data frame transmission processing.
- <8> Waits from data frame transmission until status check processing (wait time $t_{WT14} (MAX.)$).
- <9> The status frame is acquired by status check processing.
- <10> The following processing is performed according to the result of status check processing.

When the processing ends normally: Proceeds to <11>.
 When the processing ends abnormally: Abnormal termination [D]
 When a time-out error occurs: A time-out error [C] is returned.

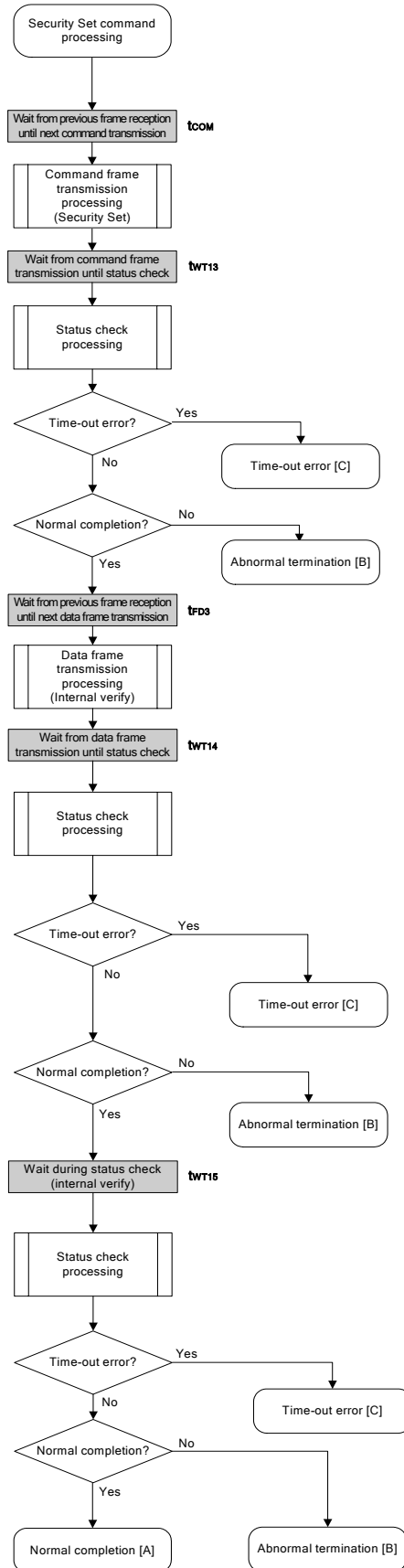
- <11> Waits until status acquisition (completion of internal verify) (wait time $t_{WT15} (MAX.)$).
- <12> The status frame is acquired by status check processing.
- <13> The following processing is performed according to the result of status check processing.

When the processing ends normally: Normal completion [A]
 When the processing ends abnormally: Abnormal termination [E]
 When a time-out error occurs: A time-out error [C] is returned.

8.15.3 Status at processing completion

Status at Processing Completion		Status Code	Description
Normal completion [A]	Normal acknowledgment (ACK)	06H	The command was executed normally and security setting was performed normally.
Abnormal termination [B]	Parameter error	05H	BLK or PAG is not 00H.
	Checksum error	07H	The checksum of the transmitted command frame or data frame does not match.
	Write error	1CH	Security data has already been set, or a write error has occurred.
	Negative acknowledgment (NACK)	15H	Command frame data is abnormal (such as invalid data length (LEN) or no ETX).
Time-out error [C]		–	The status frame was not received within the specified time.

8.15.4 Flowchart



8.15.5 Sample program

The following shows a sample program for Security Set command processing.

```

/*****/
/*                                                                 */
/* Set security flag command (CSI)                                 */
/*                                                                 */
/*****/
/* [i] u8 scf      ... Security flag data                        */
/* [r] u16         ... error code                               */
/*****/
u16      fl_csi_setscf(u8 scf)
{
    u16    rc;

    /*****/
    /*      set params                                           */
    /*****/
    fl_cmd_prm[0] = 0x00;          // "BLK" (must be 0x00)
    fl_cmd_prm[1] = 0x00;          // "PAG" (must be 0x00)
    fl_txdata_frm[0] = scf | 0b11111000;
                                   // "FLG" (upper 5bits must be '1' (to make sure))

    /*****/
    /*      send command                                         */
    /*****/
    fl_wait(tCOM);                // wait before sending command frame

    put_cmd_csi(FL_COM_SET_SECURITY, 3, fl_cmd_prm); // send "Security Set" command

    fl_wait(tWT13);                // wait

    rc = fl_csi_getstatus();        // get status frame
    switch(rc) {
        case FLC_NO_ERR:            break; // continue
        // case FLC_DFTO_ERR:       return rc; break; // case [C]
        default:                    return rc; break; // case [B]
    }

    /*****/
    /*      send data frame (security setting data)             */
    /*****/
    fl_wait(tFD3);                // wait before getting data frame

    put_dfrm_csi(1, fl_txdata_frm, true); // send data frame(Security data)

    fl_wait(tWT14);

```

```
rc = fl_csi_getstatus();          // get status frame
switch(rc) {
    case FLC_NO_ERR:              break; // continue
//   case FLC_DFTO_ERR:          return rc; break; // case [C]
    default:                      return rc; break; // case [B]
}

/*****
/*   Check internally verify   */
*****/
fl_wait(tWT15);

rc = fl_csi_getstatus(); // get status frame
// switch(rc) {
//
//   case FLC_NO_ERR:          return rc; break; // case [A]
//   case FLC_DFTO_ERR:       return rc; break; // case [C]
//   default:                  return rc; break; // case [B]
// }
return rc;
}
```

CHAPTER 9 FLASH MEMORY PROGRAMMING PARAMETER CHARACTERISTICS

“fx” in the following tables indicate the frequency of the clock input to the 78K0/Kx1+ (unit: MHz).

9.1 Flash Memory Programming Mode Setting Time

($T_A = -10$ to $+65^\circ\text{C}$, $2.7\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $2.7\text{ V} \leq AV_{REF} \leq V_{DD}$, $V_{SS} = 0\text{ V}$)

Parameter	Symbol	MIN.	TYP.	MAX.
VDD↑ to FLMD0/1↑	t _{DP}	10 ms		
FLMD0/1↑ to $\overline{\text{RESET}}\uparrow$	t _{PR}	2 ms		
Count start time from $\overline{\text{RESET}}\uparrow$ to FLMD0 ^{Note 1}	t _{RP}	78,704/fx		
Count finish time from $\overline{\text{RESET}}\uparrow$ to FLMD0 ^{Note 1}	t _{RPE}			249,952/fx
FLMD0 counter high-level/low-level width	t _{PW}	10 μs		100 μs
Wait for Read command (CSI/CSI + HS)	t _{RC}	335,934/fx		3 s
Wait for low-level data 1 (UART)	t _{R1}			3 s
Wait for low-level data 2 (UART)	t _{2C}	30,000/fx		3 s
Wait for Read command (UART)	t ₁₂	30,000/fx		3 s
Width of low-level data 1/2 ^{Note 2}	t _{L1} , t _{L2}		Note 2	
FLMD0 counter rise/fall time	–			1 μs

Notes 1. (78,704/fx + 249,952/fx)/2 is recommended as the standard value for the FLMD0 pulse input timing.

2. The low-level width is the same as the 00H data width at 9,600 bps.

9.2 Programming Characteristics

Wait	Condition	Symbol	Serial I/F	MIN.	MAX.
Between data frame transmission/reception	Data frame reception	t _{DR}	CSI	176/f _x	3 s
			UART	176/f _x	3 s
	Data frame transmission	t _{DT}	CSI	304/f _x	3 s
			UART	0 ^{Note 1}	3 s
From Status command frame reception until status frame transmission	–	t _{SF}	CSI	569/f _x	3 s
From status frame transmission until data frame transmission (1)	–	t _{FD1} ^{Note 2}	CSI	187,642/f _x + 33 μs	3 s
			UART	0 ^{Note 1}	3 s
From status frame transmission until data frame transmission (2)	Silicon signature data	t _{FD2}	CSI	10,016/f _x	3 s
	Version data			480/f _x	3 s
	–		UART	0 ^{Note 1}	3 s
From status frame transmission until data frame reception	–	t _{FD3}	CSI	192/f _x	3 s
			UART	192/f _x	3 s
From status frame transmission until command frame reception	–	t _{COM}	CSI	104/f _x	3 s
			UART	104/f _x	3 s

Notes 1. When successive reception is enabled for the programmer

2. Time for one block transmission

Remark The waits are defined as follows.

<t_{DR}, t_{FD3}, t_{COM}>

The 78K0/Kx1+ is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

The programmer must transmit the next data between the MIN. and MAX. times after completion of the previous communication.

<t_{DT}, t_{SF}, t_{FD1}, t_{FD2}>

The 78K0/Kx1+ is readied for the next communication after the MIN. time has elapsed after completion of the previous communication.

The programmer must receive the next data between the MIN. and MAX. times after completion of the previous communication.

Command	Symbol	Serial I/F	MIN.	MAX.
Reset	t _{WT0}	CSI	304/f _x	3 s
		UART	Note 1	3 s
Chip Erase	t _{WT1}	-	[78K0/KB1+] 1,444,656/f _x + 12.1 ms	[78K0/KB1+] 369,712,596/f _x + 3,089 ms
			[78K0/KC1+] 1,866,544/f _x + 12.1 ms	[78K0/KC1+] 477,715,924/f _x + 3,089 ms
			[78K0/KD1+, 78K0/KE1+, 78K0/KF1+] 3,343,152/f _x + 12.1 ms	[78K0/KD1+, 78K0/KE1+, 78K0/KF1+] 855,727,572/f _x + 3,089 ms
Block Erase	t _{WT2} ^{Note 2}	-	147,184/f _x + 12.1 ms	32,733,379/f _x + 3,089 ms
Programming	t _{WT3}	CSI	1,488/f _x + 27 μs	3 s
		UART	Note 1	3 s
	t _{WT4} ^{Note 3}	-	81,600/f _x + 25 ms	674,240/f _x + 274 ms
	t _{WT5} ^{Note 2}	CSI	363,546/f _x + 24,579 μs	436,256/f _x + 29,495 μs
UART		Note 1	436,256/f _x + 29,495 μs	
Verify	t _{WT6}	CSI	1,008/f _x	3 s
		UART	Note 1	3 s
	t _{WT7} ^{Note 3}	CSI	20,368/f _x + 27 μs	3 s
		UART	Note 1	3 s
Block Blank Check	t _{WT8} ^{Note 2}	CSI	132,368/f _x + 27 μs	158,842/f _x + 33 μs
		UART	Note 1	158,842/f _x + 33 μs
Oscillating Frequency Set	t _{WT9}	CSI	17,984/f _x	3 s
		UART	Note 1	3 s
Baud Rate Set	t _{WT10}	UART	19,200/f _x	3 s
Silicon Signature	t _{WT11}	CSI	576/f _x	3 s
		UART	Note 1	3 s
Version Get	t _{WT12}	CSI	544/f _x	3 s
		UART	Note 1	3 s
Security Set	t _{WT13}	CSI	784/f _x + 27 μs	3 s
		UART	Note 1	3 s
	t _{WT14}	-	848/f _x + 389 μs	1,018/f _x + 467 μs
	t _{WT15}	CSI	3,248/f _x + 195 μs	3,898/f _x + 234 μs
UART		Note 1	3,898/f _x + 234 μs	
Checksum	t _{WT16}	CSI	816/f _x	3 s
		UART	Note 1	3 s

Notes 1. Reception must be enabled for the programmer before command transmission.

2. Time for one block transmission
3. Time for 256-byte data transmission

Remark The waits are defined as follows.

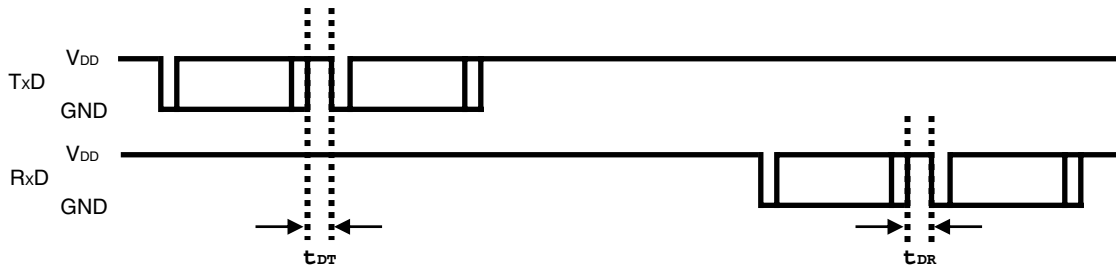
<t_{WT0} to t_{WT16}>

The 78K0/Kx1+ completes command processing between the MIN. and MAX. times.

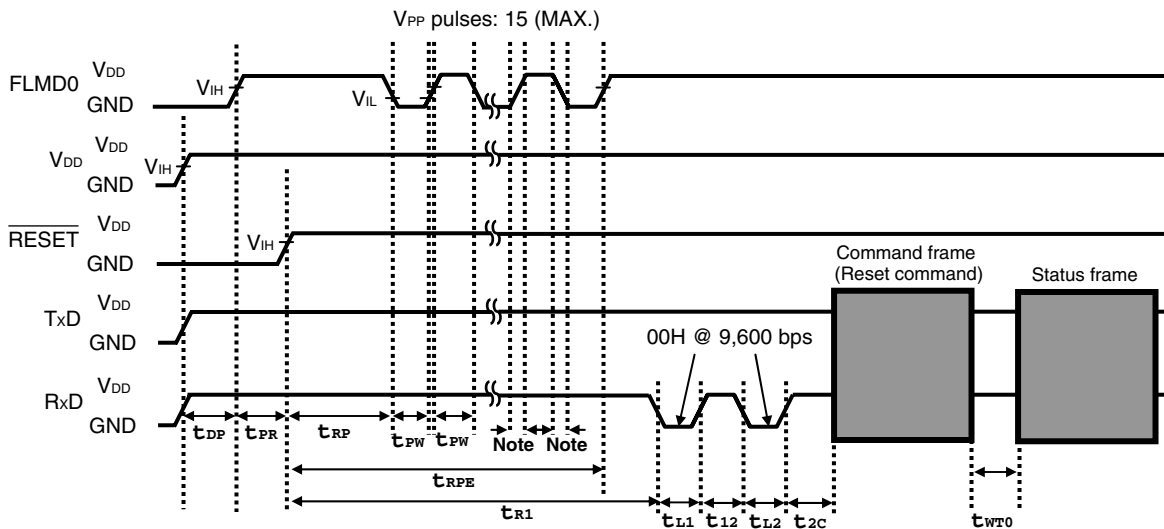
The programmer must repeat the status check until the MAX. time is elapsed.

9.3 UART Communication Mode

- Data frame

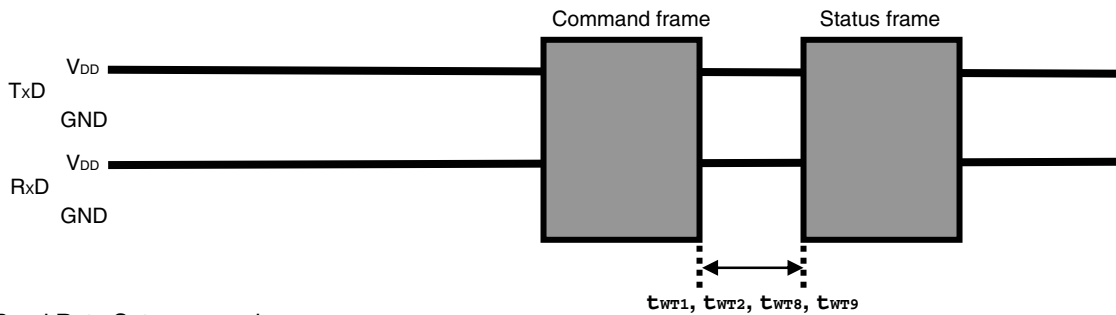


- Programming mode setting/Reset command

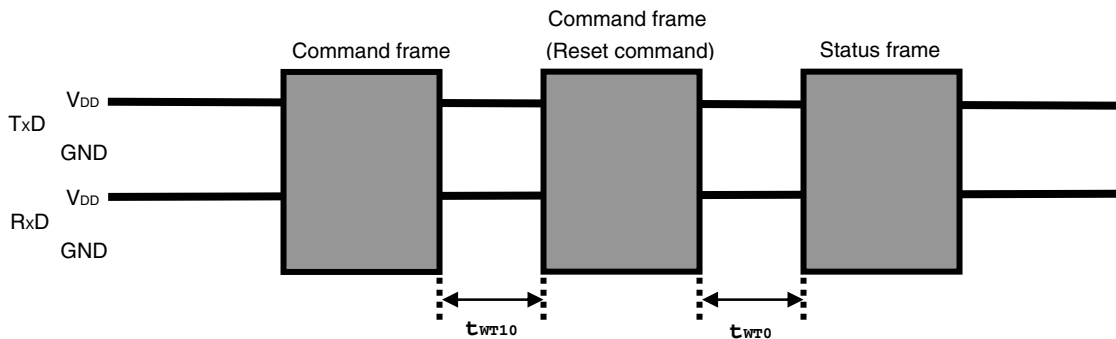


Note FLMD0 counter rise/fall time

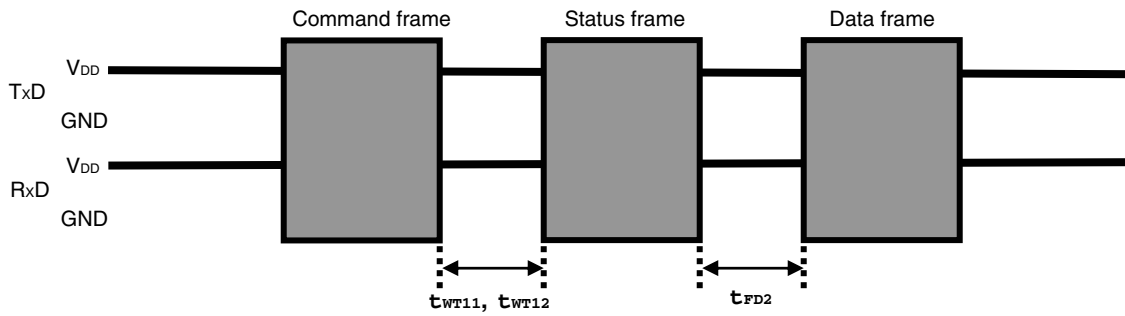
- Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command



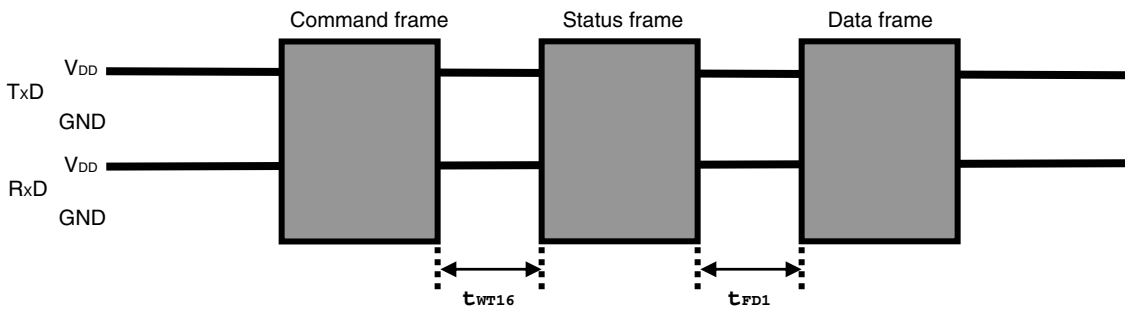
- Baud Rate Set command



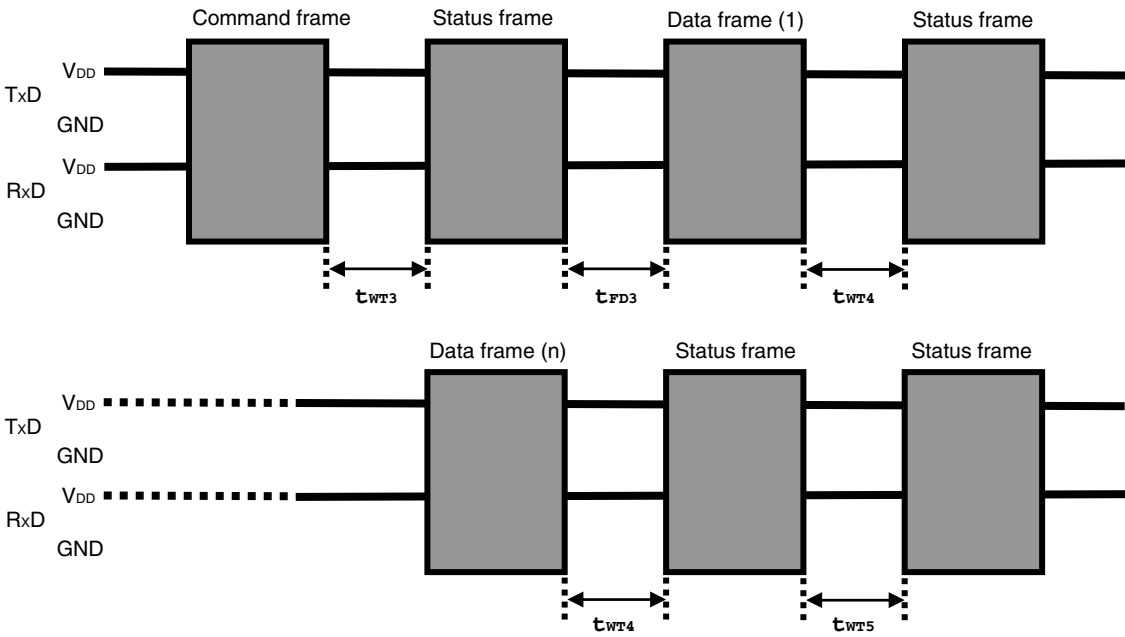
- Silicon Signature command/Version Get command



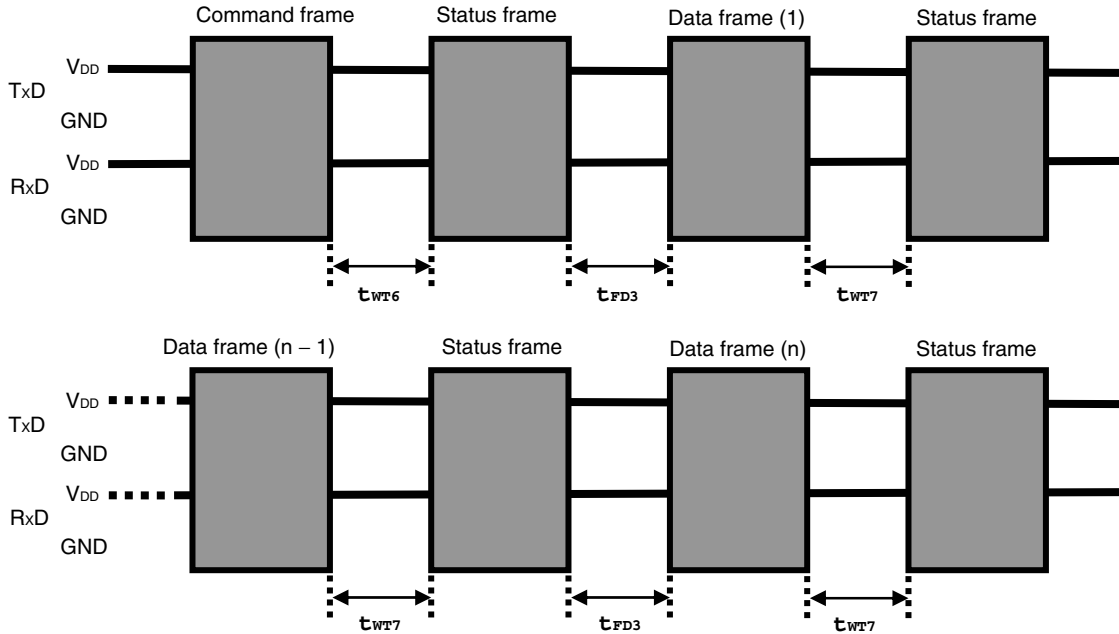
- Checksum command



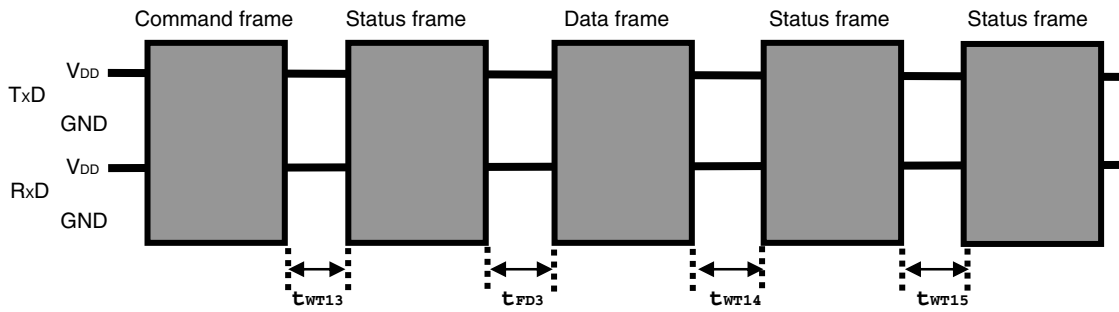
- Programming command



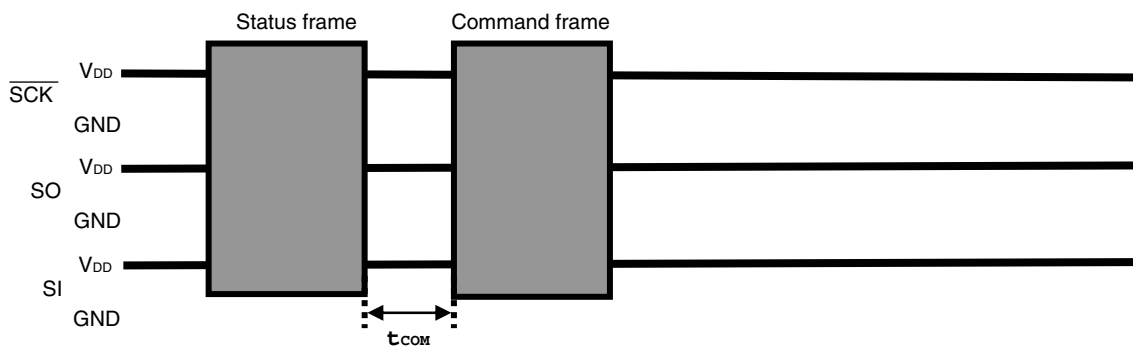
• Verify command



• Security Set command

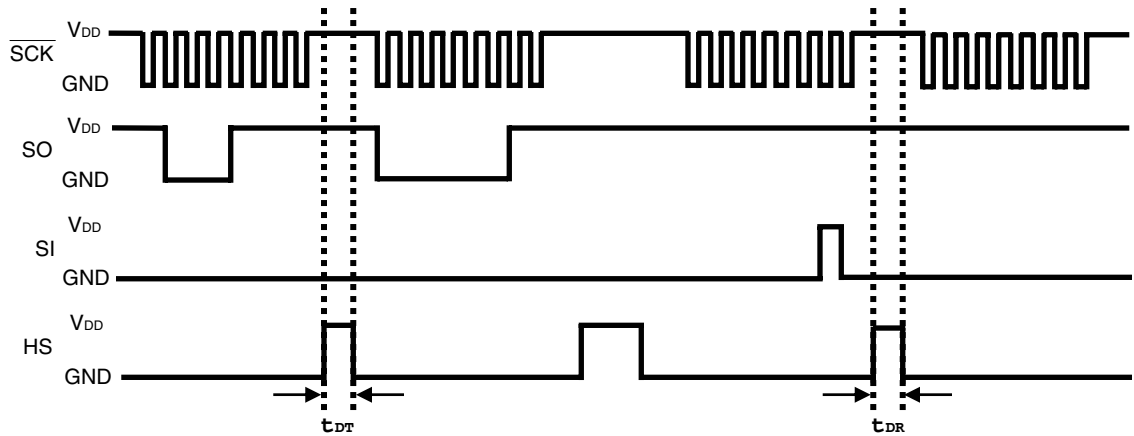


• Wait before command frame transmission

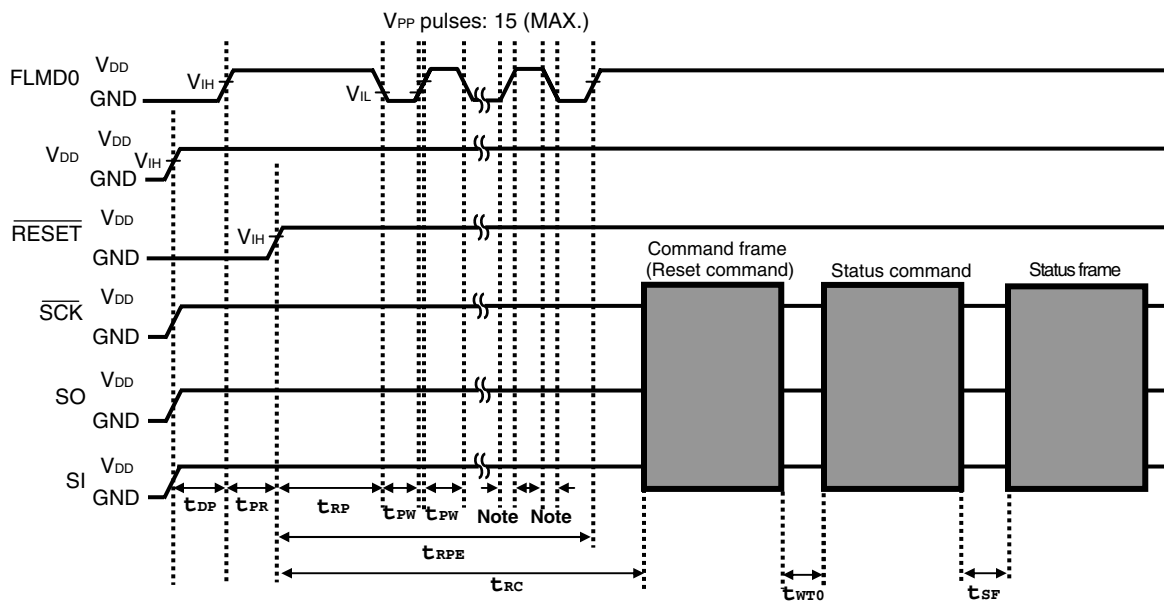


9.4 3-Wire Serial I/O Communication Mode

- Data frame

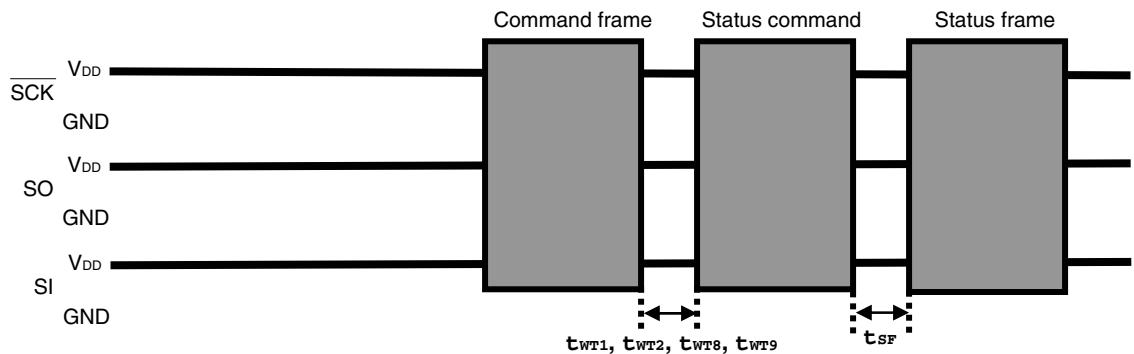


- Programming mode setting/Reset command

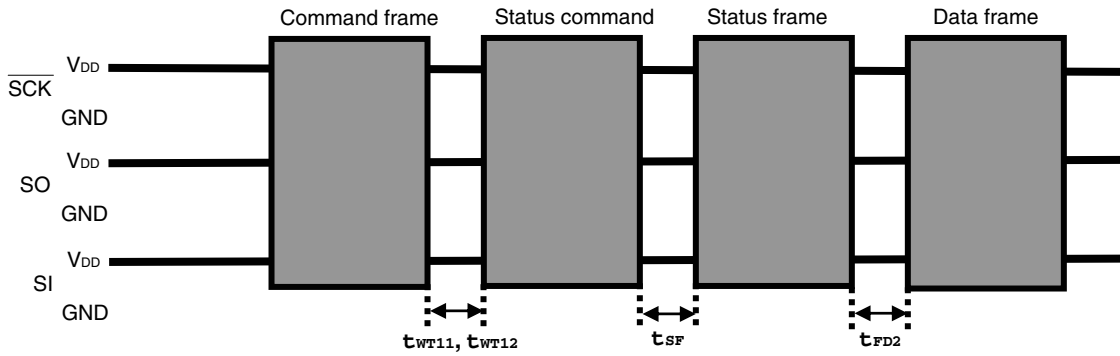


Note FLMD0 counter rise/fall time

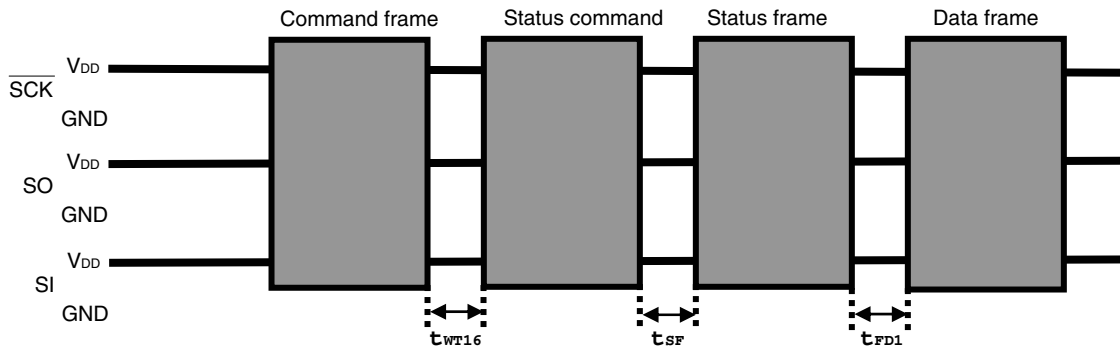
- Chip Erase command/Block Erase command/Block Blank Check command/Oscillating Frequency Set command



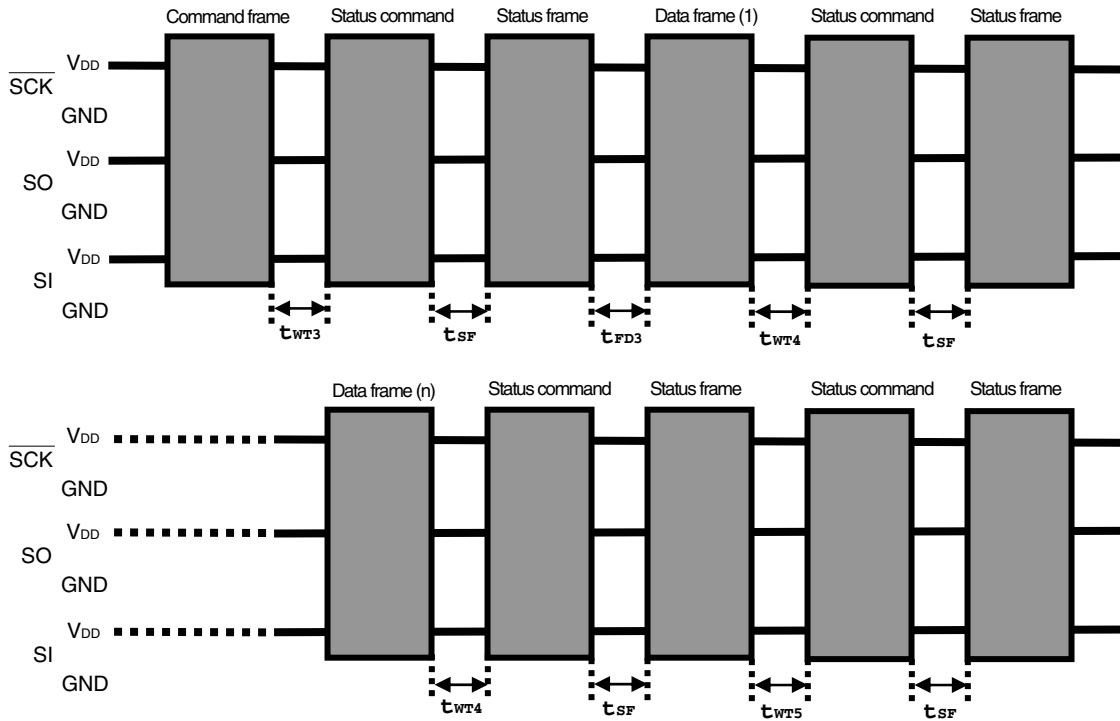
- Silicon Signature command/Version Get command



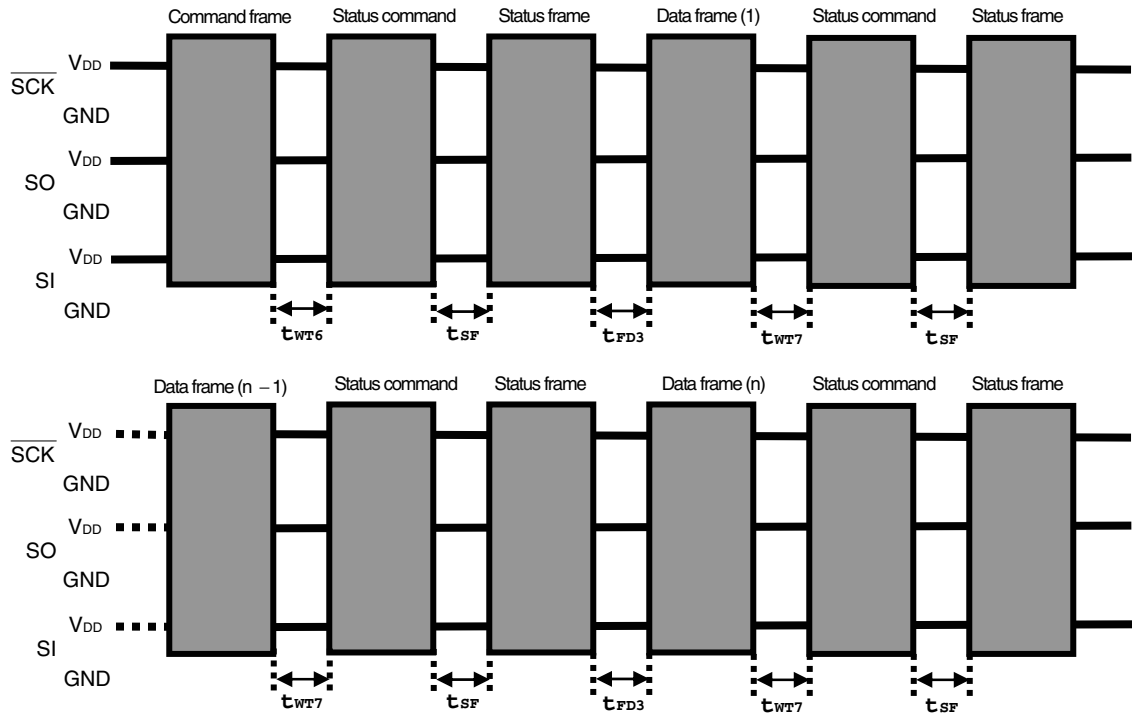
- Checksum command



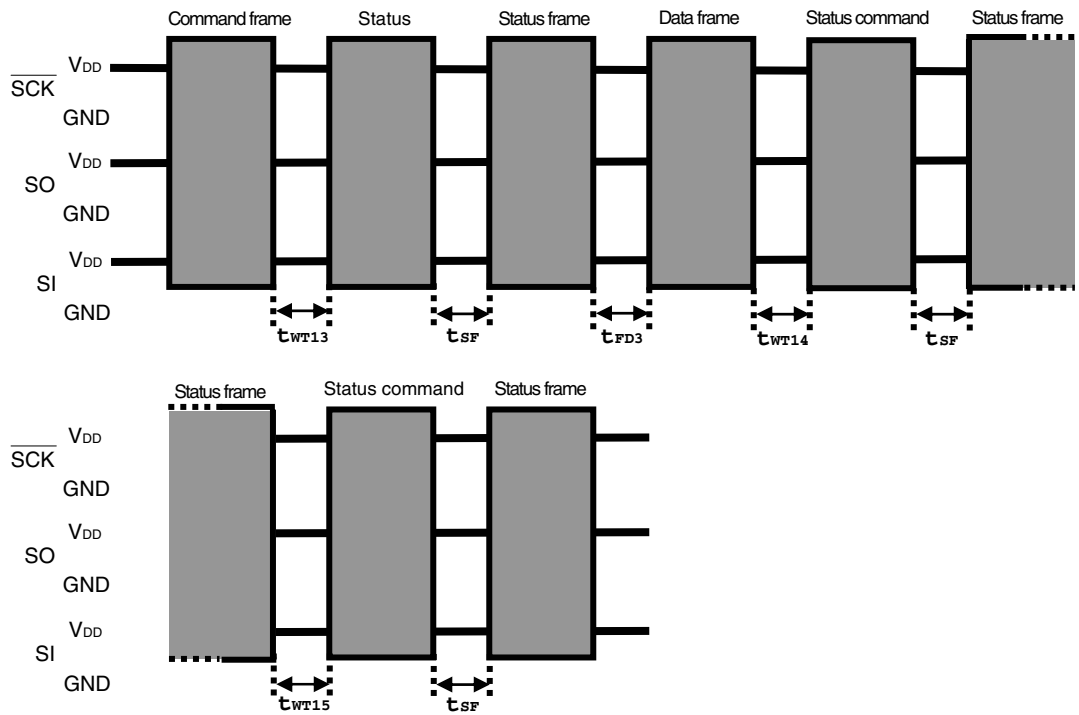
- Programming command



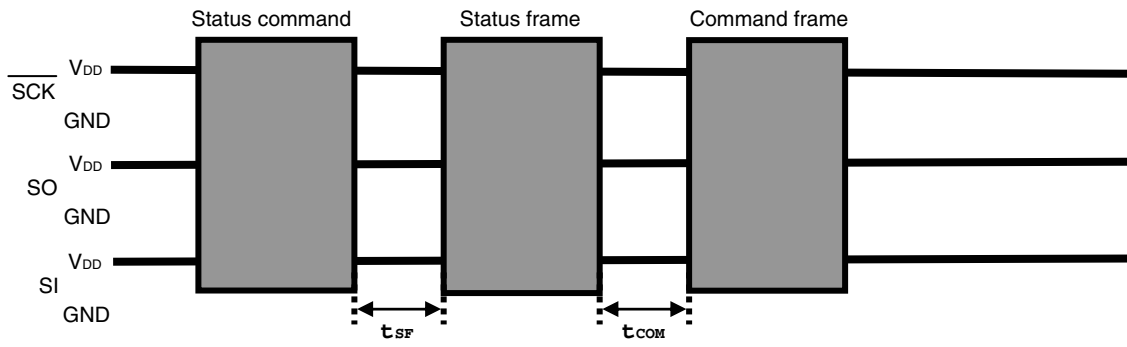
• Verify command



• Security Set command



- Wait before command frame transmission



CHAPTER 10 ELECTRICAL SPECIFICATIONS (REFERENCE)

The following specifications are for your reference only.

Be sure to refer to the latest user's manual of each product for electrical specifications when designing a programmer.

Flash Memory Programming Characteristics

($T_A = -10$ to $+65^\circ\text{C}$, $2.7\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $2.7\text{ V} \leq AV_{REF} \leq V_{DD}$, $V_{SS} = 0\text{ V}$)

Item	Symbol	Condition	MIN.	TYP.	MAX.	Unit	
Power supply current	I_{DD}	$f_{XP} = 16\text{ MHz}$, $V_{DD} = 5.5\text{ V}$	78K0/KB1+			30.5	mA
			78K0/KC1+, 78K0/KD1+			32	mA
			78K0/KE1+, 78K0/KF1+			35	mA

High-Speed System Clock (Crystal/Ceramic) Oscillator Characteristics

 (T_A = -10 to +65°C, 2.7 V ≤ V_{DD} ≤ 5.5 V, 2.7 V ≤ AV_{REF} ≤ V_{DD}, V_{SS} = 0 V)

Resonator	Recommended Circuit	Parameter	Conditions	MIN.	TYP.	MAX.	Unit
Ceramic resonator		Oscillation frequency (f _{XP})	4.0 V ≤ V _{DD} ≤ 5.5 V	2.0		16	MHz
			3.5 V ≤ V _{DD} < 4.0 V	2.0		10	
			3.0 V ≤ V _{DD} < 3.5 V	2.0		8.38	
			2.5 V ≤ V _{DD} < 3.0 V	2.0		5.0	
Crystal resonator		Oscillation frequency (f _{XP})	4.0 V ≤ V _{DD} ≤ 5.5 V	2.0		16	MHz
			3.5 V ≤ V _{DD} < 4.0 V	2.0		10	
			3.0 V ≤ V _{DD} < 3.5 V	2.0		8.38	
			2.5 V ≤ V _{DD} < 3.0 V	2.0		5.0	
External clock		X1 input frequency (f _{XP})	4.0 V ≤ V _{DD} ≤ 5.5 V	2.0		16	MHz
			3.5 V ≤ V _{DD} < 4.0 V	2.0		10	
			3.0 V ≤ V _{DD} < 3.5 V	2.0		8.38	
			2.5 V ≤ V _{DD} < 3.0 V	2.0		5.0	
		X1 input high-/low-level width (t _{xPH} , t _{xPL})	4.0 V ≤ V _{DD} ≤ 5.5 V	30		250	ns
			3.5 V ≤ V _{DD} < 4.0 V	46		250	
			3.0 V ≤ V _{DD} < 3.5 V	56		250	
			2.5 V ≤ V _{DD} < 3.0 V	96		250	

Caution When using the high-speed system clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.

- Keep the wiring length as short as possible.
- Do not cross the wiring with the other signal lines.
- Do not route the wiring near a signal line through which a high fluctuating current flows.
- Always make the ground point of the oscillator capacitor the same potential as V_{SS}.
- Do not ground the capacitor to a ground pattern through which a high current flows.
- Do not fetch signals from the oscillator.

DC Characteristics ($T_A = -10$ to $+65^\circ\text{C}$, $2.7\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $2.7\text{ V} \leq AV_{REF} \leq V_{DD}$, $V_{SS} = 0\text{ V}$) (1/3)

(1) 78K0/KB1+, 78K0/KE1+, 78K0/KF1+

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Output current, high	I_{OH}	Per pin	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			-5	mA
		Total of $\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			-25	mA
		All pins	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$			-10	mA
Output current, low	I_{OL}	Per pin	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			10	mA
		Total of $\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			30	mA
		All pins	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$			10	mA
Input voltage, high	V_{IH1}	SO10, TxD6, HS		$0.7V_{DD}$		V_{DD}	V
	V_{IH2}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, RxD6, FLMD1, $\overline{\text{RESET}}$		$0.8V_{DD}$		V_{DD}	V
	V_{IH3}	X1, X2		$V_{DD} - 0.5$		V_{DD}	V
Input voltage, low	V_{IL1}	SO10, TxD6, HS		0		$0.3V_{DD}$	V
	V_{IL2}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, RxD6, FLMD1, $\overline{\text{RESET}}$		0		$0.2V_{DD}$	V
	V_{IL3}	X1, X2		0		0.4	V
Output voltage, high	V_{OH}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1 Total $I_{OH} = -25\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OH} = -5\text{ mA}$	$V_{DD} - 1.0$			V
		$I_{OH} = -100\text{ }\mu\text{A}$	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$	$V_{DD} - 0.5$			V
Output voltage, low	V_{OL}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1 Total $I_{OL} = 30\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OL} = 10\text{ mA}$			1.3	V
		$I_{OL} = 400\text{ }\mu\text{A}$	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$			0.4	V
Input leakage current, high	I_{LH1}	$V_i = V_{DD}$	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\text{RESET}}$			3	μA
	I_{LH2}	$V_i = V_{DD}$	X1, X2 ^{Note}			20	μA
Input leakage current, low	I_{LIL1}	$V_i = 0\text{ V}$	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, $\overline{\text{RESET}}$			-3	μA
	I_{LIL2}		X1, X2 ^{Note}			-20	μA
Output leakage current, high	I_{LOH}	$V_o = V_{DD}$				3	μA
Output leakage current, low	I_{LOL}	$V_o = 0\text{ V}$				-3	μA
Pull-up resistor	R_L	$V_i = 0\text{ V}$		10	30	100	k Ω
FLMD0 supply voltage	F _{lmd}	In normal operation mode		0		$0.2V_{DD}$	V

Note When inverted input level of X1 is applied to X2

DC Characteristics ($T_A = -10$ to $+65^\circ\text{C}$, $2.7\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $2.7\text{ V} \leq AV_{REF} \leq V_{DD}$, $V_{SS} = 0\text{ V}$) (2/3)

(2) 78K0/KC1+

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Output current, high	I_{OH}	Per pin	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			-5	mA
		Total of $\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , SO10 , TxD6 , RxD6 , HS	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			-25	mA
		FLMD1	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			-25	mA
		All pins	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$			-10	mA
Output current, low	I_{OL}	Per pin	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			10	mA
		Total of $\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , SO10 , TxD6 , RxD6 , HS	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			30	mA
		FLMD1	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$			30	mA
		All pins	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$			10	mA
Input voltage, high	V_{IH1}	SO10 , TxD6 , HS		$0.7V_{DD}$		V_{DD}	V
	V_{IH2}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , RxD6 , FLMD1 , RESET		$0.8V_{DD}$		V_{DD}	V
	V_{IH3}	X1, X2		$V_{DD} - 0.5$		V_{DD}	V
Input voltage, low	V_{IL1}	SO10 , TxD6 , HS		0		$0.3V_{DD}$	V
	V_{IL2}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , RxD6 , FLMD1 , RESET		0		$0.2V_{DD}$	V
	V_{IL3}	X1, X2		0		0.4	V
Output voltage, high	V_{OH}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , SO10 , TxD6 , RxD6 , HS Total $I_{OH} = -25\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OH} = -5\text{ mA}$	$V_{DD} - 1.0$			V
		FLMD1 $I_{OH} = -25\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OH} = -5\text{ mA}$	$V_{DD} - 1.0$			V
		$I_{OH} = -100\text{ }\mu\text{A}$	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$	$V_{DD} - 0.5$			V
Output voltage, low	V_{OL}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , SO10 , TxD6 , RxD6 , HS Total $I_{OL} = 30\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OL} = 10\text{ mA}$			1.3	V
		FLMD1 $I_{OL} = 30\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OL} = 10\text{ mA}$			1.3	V
		$I_{OL} = 400\text{ }\mu\text{A}$	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$			0.4	V
Input leakage current, high	I_{LIH1}	$V_i = V_{DD}$	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , SO10 , TxD6 , RxD6 , HS , FLMD1 , RESET			3	μA
	I_{LIH2}	$V_i = V_{DD}$	X1, X2 ^{Note}			20	μA
Input leakage current, low	I_{LIL1}	$V_i = 0\text{ V}$	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0 , SO10 , TxD6 , RxD6 , HS , FLMD1 , RESET			-3	μA
	I_{LIL2}		X1, X2 ^{Note}			-20	μA
Output leakage current, high	I_{LOH}	$V_o = V_{DD}$				3	μA
Output leakage current, low	I_{LOL}	$V_o = 0\text{ V}$				-3	μA
Pull-up resistor	R_L	$V_i = 0\text{ V}$		10	30	100	k Ω
FLMD0 supply voltage	Flmd	In normal operation mode		0		$0.2V_{DD}$	V

Note When inverted input level of X1 is applied to X2

DC Characteristics ($T_A = -10$ to $+65^\circ\text{C}$, $2.7\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $2.7\text{ V} \leq AV_{REF} \leq V_{DD}$, $V_{SS} = 0\text{ V}$) (3/3)

(3) 78K0/KD1+

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit	
Output current, high	I_{OH}	Per pin	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$		-5	mA	
		Total of $\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$		-25	mA	
		Total of HS, FLMD1	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$		-25	mA	
		All pins	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$		-10	mA	
Output current, low	I_{OL}	Per pin	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$		10	mA	
		Total of $\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$		30	mA	
		Total of HS, FLMD1	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$		30	mA	
		All pins	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$		10	mA	
Input voltage, high	V_{IH1}	SO10, TxD6, HS		$0.7V_{DD}$	V_{DD}	V	
	V_{IH2}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, RxD6, FLMD1, RESET		$0.8V_{DD}$	V_{DD}	V	
	V_{IH3}	X1, X2		$V_{DD} - 0.5$	V_{DD}	V	
Input voltage, low	V_{IL1}	SO10, TxD6, HS		0	$0.3V_{DD}$	V	
	V_{IL2}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, RxD6, FLMD1, RESET		0	$0.2V_{DD}$	V	
	V_{IL3}	X1, X2		0	0.4	V	
Output voltage, high	V_{OH}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6 Total $I_{OH} = -25\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OH} = -5\text{ mA}$	$V_{DD} - 1.0$		V	
		HS, FLMD1 Total $I_{OH} = -25\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OH} = -5\text{ mA}$	$V_{DD} - 1.0$		V	
		$I_{OH} = -100\text{ }\mu\text{A}$	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$	$V_{DD} - 0.5$		V	
Output voltage, low	V_{OL}	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6 Total $I_{OL} = 30\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OL} = 10\text{ mA}$		1.3	V	
		HS, FLMD1 Total $I_{OL} = 30\text{ mA}$	$4.0\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $I_{OL} = 10\text{ mA}$		1.3	V	
		$I_{OL} = 400\text{ }\mu\text{A}$	$2.7\text{ V} \leq V_{DD} < 4.0\text{ V}$		0.4	V	
Input leakage current, high	I_{LH1}	$V_i = V_{DD}$	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, RESET		3	μA	
	I_{LH2}	$V_i = V_{DD}$	X1, X2 ^{Note}		20	μA	
Input leakage current, low	I_{LIL1}	$V_i = 0\text{ V}$	$\overline{\text{SCK10/TxD0}}$, SI10/RxD0, SO10, TxD6, RxD6, HS, FLMD1, RESET		-3	μA	
	I_{LIL2}		X1, X2 ^{Note}		-20	μA	
Output leakage current, high	I_{LOH}	$V_o = V_{DD}$			3	μA	
Output leakage current, low	I_{LOL}	$V_o = 0\text{ V}$			-3	μA	
Pull-up resistor	R_L	$V_i = 0\text{ V}$		10	30	100	k Ω
FLMD0 supply voltage	F _{lmd}	In normal operation mode		0	$0.2V_{DD}$	V	

Note When inverted input level of X1 is applied to X2

AC Characteristics

(1) Basic operation ($T_A = -10$ to $+65^\circ\text{C}$, $2.7\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $2.7\text{ V} \leq AV_{REF} \leq V_{DD}$, $V_{SS} = 0\text{ V}$)

Item	Symbol	Condition	MIN.	TYP.	MAX.	Unit
$\overline{\text{RESET}}$ input low-level width	t_{RSL}		10			μs

(2) Serial interface ($T_A = -10$ to $+65^\circ\text{C}$, $2.7\text{ V} \leq V_{DD} \leq 5.5\text{ V}$, $2.7\text{ V} \leq AV_{REF} \leq V_{DD}$, $V_{SS} = 0\text{ V}$)

(a) UART mode (UART6, dedicated baud rate generator output)

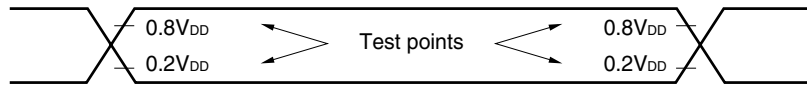
Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Transfer rate					312.5	kbps

(b) 3-wire serial I/O mode (slave mode, $\overline{\text{SCK10}}$... external clock input)

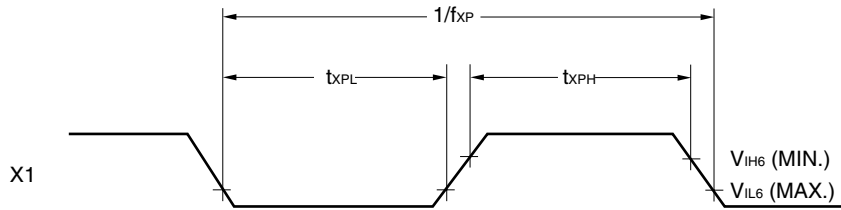
Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
$\overline{\text{SCK10}}$ cycle time	t_{KV2}		400			ns
$\overline{\text{SCK10}}$ high-/low-level width	t_{KH2} , t_{KL2}		$t_{\text{KV2}}/2$			ns
SI10 setup time (to $\overline{\text{SCK10}}\uparrow$)	t_{SIK2}		80			ns
SI10 hold time (from $\overline{\text{SCK10}}\uparrow$)	t_{KSI2}		50			ns
Delay time from $\overline{\text{SCK10}}\downarrow$ to SO10 output	t_{KSO2}	$C = 100\text{ pF}^{\text{Note}}$			120	ns

Note C is the load capacitance of the SO10 output line.

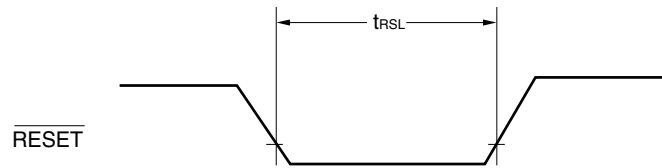
AC Timing Test Points (Excluding X1)



Clock Timing

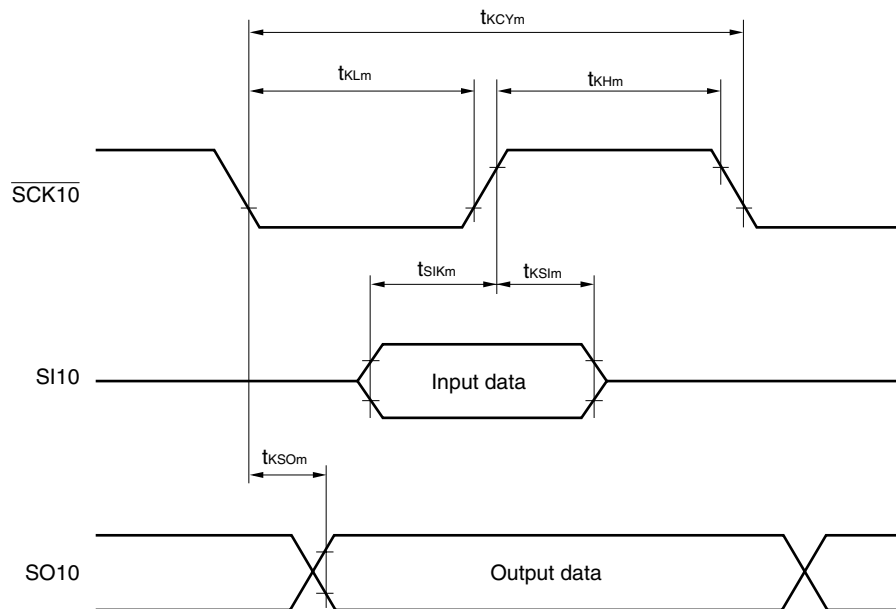


RESET Input Timing



Serial Transfer Timing

3-wire serial I/O mode:



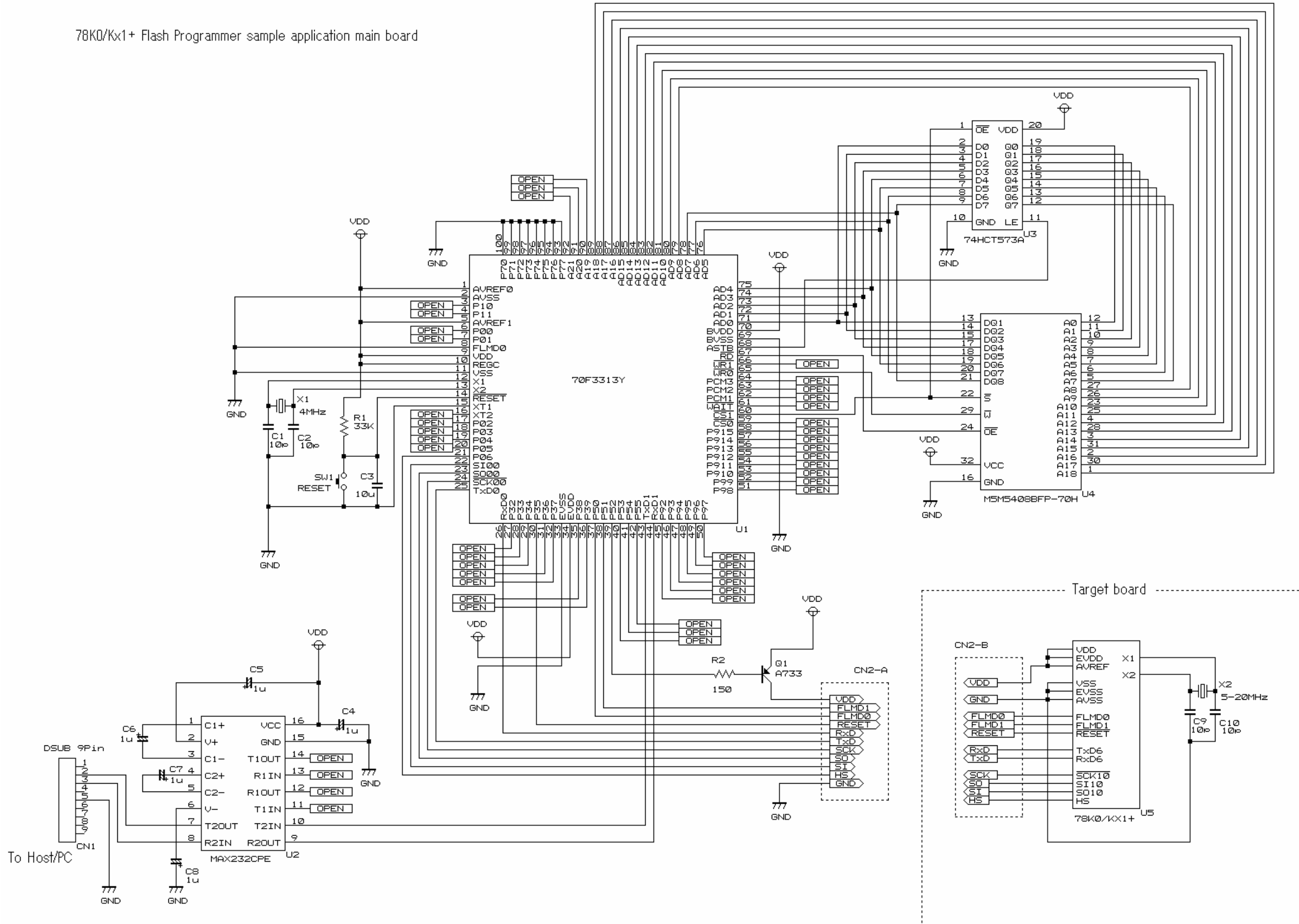
Remark $m = 1, 2$

APPENDIX A CIRCUIT DIAGRAM (REFERENCE)

Figure A-1 shows a circuit diagram of the programmer and the 78K0/Kx1+, for reference.

Figure A-1. Reference Circuit Diagram of Programmer and 78K0/Kx1+

78K0/Kx1+ Flash Programmer sample application main board



*For further information,
please contact:*

NEC Electronics Corporation

1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
<http://www.necel.com/>

[America]

NEC Electronics America, Inc.

2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
<http://www.am.necel.com/>

[Europe]

NEC Electronics (Europe) GmbH

Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
<http://www.eu.necel.com/>

Hanover Office

Podbielskistrasse 164
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office

Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office

Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch

Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française

9, rue Paul Dautier, B.P. 52180
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España

Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial

Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana

Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands

Limburglaan 5
5616 HR Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd

7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
TEL: 010-8235-1155
<http://www.cn.necel.com/>

NEC Electronics Shanghai Ltd.

Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai P.R. China P.C:200120
Tel: 021-5888-5400
<http://www.cn.necel.com/>

NEC Electronics Hong Kong Ltd.

12/F., Cityplaza 4,
12 Taikoo Wan Road, Hong Kong
Tel: 2886-9318
<http://www.hk.necel.com/>

Seoul Branch

11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737

NEC Electronics Taiwan Ltd.

7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-2719-2377

NEC Electronics Singapore Pte. Ltd.

238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
<http://www.sg.necel.com/>